NAVAL POSTGRADUATE SCHOOL
Monterey, California

THESIS

CLASSIFICATION AND ANALYSIS OF LOW PROBABILITY OF INTERCEPT RADAR SIGNALS USING IMAGE PROCESSING

by

Christer Persson

September 2003

Thesis Advisor:	Phillip E. Pace
Co-Advisor:	D. Curtis Schleher

REPORT DOCUMENTATION PAGE		*Form Approved OMB No. 0704-0188*

1. AGENCY USE ONLY *(Leave blank)*	**2. REPORT DATE** September 2003	**3. REPORT TYPE AND DATES COVERED** Master's Thesis
4. TITLE AND SUBTITLE: Classification and Analysis of Low Probability of Intercept Radar Signals Using Image Processing		**5. FUNDING NUMBERS**
6. AUTHOR(S) Christer Persson		
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Center for Joint Services Electronic Warfare, Naval Postgraduate School, Monterey, CA 93943-5000		**8. PERFORMING ORGANIZATION REPORT NUMBER**
9. SPONSORING /MONITORING AGENCY NAME(S) AND ADDRESS(ES) N/A		**10. SPONSORING/MONITORING AGENCY REPORT NUMBER**
11. SUPPLEMENTARY NOTES The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government.		
12a. DISTRIBUTION/AVAILABILITY STATEMENT Approved for public release; distribution is unlimited		**12b. DISTRIBUTION CODE**

13. ABSTRACT *(maximum 200 words)*

The characteristic of low probability of intercept (LPI) radar makes it difficult to intercept with conventional signal intelligence methods so new interception methods need to be developed. This thesis initially describes a simulation of a polytime phase–coded LPI signal. The thesis then introduces a method for classification of LPI radar signals. The method utilizes a parallel tree structure with three separate "branches" to exploit the image representation formed by three separate detection methods. Each detection method output is pre–processed and features are extracted using image processing. After processing the images, they are each fed into three separate neural networks to be classified. The classification output of each neural network is then combined and fed into a fourth neural network performing the final classification. The outcome of testing shows only 53%, which might be the result of the image representation of the detection methods not being distinct enough, the pre –processing/feature extraction not being able to extract relevant information or the neural networks not being properly trained. The thesis concludes with a brief discussion about a suitable method for image processing to extract significant parameters from a LPI signal.

14. SUBJECT TERMS Signal Processing, Image Processing, LPI, LPI Radar Signals, Classification			**15. NUMBER OF PAGES** 149
			16. PRICE CODE
17. SECURITY CLASSIFICATION OF REPORT Unclassified	**18. SECURITY CLASSIFICATION OF THIS PAGE** Unclassified	**19. SECURITY CLASSIFICATION OF ABSTRACT** Unclassified	**20. LIMITATION OF ABSTRACT** UL

NSN 7540-01-280-5500 Standard Form 298 (Rev. 2-89)
Prescribed by ANSI Std. 239-18

THIS PAGE INTENTIONALLY LEFT BLANK

CLASSIFICATION AND ANALYSIS OF LOW PROBABILITY OF INTERCEPT RADAR SIGNALS USING IMAGE PROCESSING

Christer N. E. Persson
Lieutenant Colonel, Swedish Air Force
BSSE, Swedish National Defence College, 1999

Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE IN SYSTEMS ENGINEERING
and
MASTER OF SCIENCE IN ENGINEERING SCIENCE
(ELECTRICAL ENGINEERING)

from the

NAVAL POSTGRADUATE SCHOOL
September 2003

Author:	Christer N. E. Persson
Approved by:	Phillip E. Pace Thesis Advisor
	D. Curtis Schleher Co-Advisor
	Dan C. Boger Chairman, Department of Information Sciences
	John P. Powers Chairman, Department of Electrical and Computer Engineering

THIS PAGE INTENTIONALLY LEFT BLANK

ABSTRACT

The characteristic of low probability of intercept (LPI) radar makes it difficult to intercept with conventional signal intelligence methods so new interception methods need to be developed. This thesis initially describes a simulation of a polytime phase–coded LPI signal. The thesis then introduces a method for classification of LPI radar signals. The method utilizes a parallel tree structure with three separate "branches" to exploit the image representation formed by three separate detection methods. Each detection method output is pre–processed and features are extracted using image processing. After processing the images, they are each fed into three separate neural networks to be classified. The classification output of each neural network is then combined and fed into a fourth neural network performing the final classification. The outcome of testing shows only 53%, which might be the result of the image representation of the detection methods not being distinct enough, the pre –processing/feature extraction not being able to extract relevant information or the neural networks not being properly trained. The thesis concludes with a brief discussion about a suitable method for image processing to extract significant parameters from a LPI signal.

THIS PAGE INTENTIONALLY LEFT BLANK

TABLE OF CONTENTS

LIST OF FIGURES

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF TABLES

THIS PAGE INTENTIONALLY LEFT BLANK

ACKNOWLEDGMENTS

Initially I would like to thank Professor Phillip E. Pace for the interest he showed in this thesis work and me. I further would like to express my gratitude to my son Andreas, and to Malin, that they joined me in the beginning of this journey to the other side of the world. I especially would like to express my gratitude to Jamie for her support during the last couple of months. You gave me strength!

I also would like to thank the Swedish Armed Forces for sending me to this beautiful place and last but not least, I would like to thank all of my new friends from all over the world that have made the time here an adventure.

THIS PAGE INTENTIONALLY LEFT BLANK

EXECUTIVE SUMMARY

Today radar systems face a variety of threats on the battlefield. In order to survive, different radar techniques have been developed; among these techniques low probability of intercept (LPI) waveforms are especially interesting. Those waveforms are typically continuous wave (CW) waveforms. The reason for this is that it gives the radar a possibility to detect a target without the risk of revealing itself. To obtain LPI characteristics, the LPI radar modifies its attributes to make interception of the radar difficult by Electronic Support (ES) systems or Electronic Intelligence (ELINT) systems.

Recent work has focused on how to detect a LPI radar early enough to counter an eventual attack. Unfortunately interception alone does not solve the problem. To effectively counter LPI radar (or any radar), classification of the radar type and the connection of this radar–class to a platform and/or a weapon system must be accomplished. This then serves as a basis for decisions about suitable countermeasures. This thesis focuses on the classification of LPI radar and on the extraction of significant parameters describing the radar.

The first part of this thesis work describes the LPI radar signals that the classification method intends to analyze. To limit the amount of work, only five different types of LPI signals are used. The five signals are selected to represent typical groups of LPI signals that exist. They are also chosen since similarities exist between the image representations that are produced by the time–frequency/bifrequency detection methods that are used. Each signal is described using the power spectrum magnitude and the periodic ambiguity function (PAF). The PAF is a powerful tool when analyzing LPI CW signals. A brief introduction of the PAF is given in the text.

The first signal type that is described is a Binary Phase Shift Keying signal (BPSK). Although not a LPI signal, it is often included for test purposes. The next signal type is the frequency modulated continuous wave (FMCW) signal that is chosen since it is one of the first LPI signals and still widely used. The third and the fourth signals are two phase–coded signals. The third is a Frank–coded signal derived from a linear frequency modulated waveform. The fourth phase–coded signal is the polyphase coded signal type 4 (P4). As with the Frank–code, the P4 is derived from a linear frequency modulated waveform showing great similarity with the Frank-code. The fifth signal type used is the polytime phase–coded signal. This code is described in more detail in the text since part of this thesis contribution includes developing a simulation of this signal using MATLAB. The code for the simulation is included as an appendix.

The thesis then describes the three detection methods generating the images used as input to the classification algorithm. The detection methods are the previously developed Wigner–Ville (WD) distribution method, the quadrature mirror filter bank (QMFB) method and the cyclostationary processing (CYCL) method. Each of the three methods generates a 2-D description of the signal. The WD and the QMFB represent the signal in a time–frequency plane while the CYCL represents the signal in a frequency—cycle frequency plane (bifrequency). The images are pre–processed to extract the important features, which are then used as inputs to the classification algorithm. The pre–processing/feature extraction is based on image processing being applied to the images. The main goal is to reduce the number of pixels in the images and still maintain the characteristics of the signal representation. The reduction in size is needed to make the classification process computationally acceptable.

The classification of the reduced images is performed by three separate "pipes," each processing an image generated by one of the detection methods. In each "pipe" there is a feed forward neural network that performs the classification. Each neural network is trained with a mixture of "pure" signals and "noisy" signals, all after being processed by the pre–processing/feature extraction procedure developed for that "pipe." The outputs from the three different neural networks, in the form of column-vectors, are combined into one combined column-vector then fed into a fourth (combining) feed forward neural network that decides the final classification.

The results from testing the classification algorithm are not as good as originally expected. A variation in the modulation of the signal or added noise to the signal makes it hard for the algorithm to correctly classify the signal. All tests resulted in the classification using the WD image representation performing equal or better than the combining neural network did. The reasons for this might be several. It might be that the pre–processing /feature extraction of the WD is especially good. It might also be that the WD image representation gives a more distinct image variation when the signal is changed. In particular, the WD contains frequency cross–terms that seem to make the image representation of the signal more distinct when the signal is varied. There is also a possibility that more training is needed for the fourth neural network to make correct classification even if the signals are varied.

The thesis concludes by describing the proposed method of extracting significant parameters from the detected signal after identification by the classification algorithm. The method, based on image processing, uses the best image representation available from the three detection methods. To choose which image representation to use, a table lookup is performed based on the classification result. The parameters proposed being extracted are the carrier frequency, the bandwidth and the period. From these three parameters several other parameters are possible to obtain by calculation.

THIS PAGE INTENTIONALLY LEFT BLANK

I. INTRODUCTION

A. LOW PROBABILITY OF INTERCEPT RADAR

Today radar systems face a variety of threats on the battlefield. In order to survive, different radar techniques have been developed; among these techniques low probability of intercept (LPI) waveforms are especially interesting. Those waveforms are typically continuous wave (CW) waveforms. The reason for this is that it gives the radar a possibility to detect a target without the risk of revealing itself. To obtain LPI characteristics, the LPI radar modifies its attributes to make interception of the radar difficult by Electronic Support (ES) systems or Electronic Intelligence (ELINT) systems [1].

The common attributes that characterize LPI radar are

- Low average power
- Radiated energy spread over a wide angular region, long time interval and wide frequency band.
- Continuous wave (CW) radiation with a large time-bandwidth product.
- Low side-lobe transmit antenna adapted to the carrier frequency.
- Reduced radar noise temperature and overall radar loss. [1]

To spread the signal bandwidth the transmitter uses sophisticated frequency and phase modulation. The receiver makes use of an appropriate matched filter resulting in a final radar performance similar to that of traditional pulsed radar radiating at a much higher average power. [2]

Recent work has focused on how to detect a LPI radar early enough to counter an eventual attack. Unfortunately interception alone does not solve the problem. To effectively counter LPI radar (or any radar), classification of the radar type and the connection of this radar–class to a platform and/or a weapon system must be accomplished. This then serves as a basis for decisions about suitable countermeasures.

This thesis focuses on the classification of LPI radar and on the extraction of significant parameters describing the radar. Once the classification is done and the parameters extracted, the process of determining the radar type and associated weapons can be made.

Table 1 gives examples of current LPI radars and their use together with their manufacturer. The table also shows what type of waveform the radar uses if known. These waveforms are described in Chapter II. Table 2 introduces various definitions frequently used in LPI systems.

Table 1 Current LPI Radar Systems under Production (From [3].).

Developer	System	Technique used	LPI Use
Honeywell	HG9550	Frequency agility	Radar altimeter
Navair	GRA-2000	-	Tri–service radar altimeter
NavCom Defense Electronics	AN/APN-232	FMCW	Combined altitude radar
Thompson CSF	AHV-2100	-	Radar altimeter
BAE	AD1990	Frequency hopping	Radar altimeter
Saab Bofors	Pilot Mk 1,2,3	Fast frequency hopping	Surveillance, navigation
Signaal's	Scout	FMCW	Surveillance, navigation
Textron Systems	AN/SPN-46		Precision approach, landing
Signaal's	Smart-L	-	Surveillance
Telephonics	AN/APS-147	Frequency agility	Enhanced search, target designation
Sierra Nevada	TALS	-	Tactical automatic landing system
Ericsson	Eagle	-	Fire control
Northrop Grumman	AN/APG-77	Frequency agility	Multi-mode tactical radar for F-22
Raytheon	AN/APG-70	Frequency agility	Multi-mode tactical radar for F-15E
TI	LANTIRN	-	Terrain following radar F-16C/D, F-15E, F-14
Raytheon	AN/APG-181	-	Multi-mode radar for B-2
Chinese	JY-17A	-	Battlefield surveillance radar
Raytheon	MRSR	-	Target acquisition and tracking
Saab Dynamics	RBS-15MR	-	Radar guided air-to-surface missile

Table 2 Definition of Common Terminology in the LPI Radar Field (From [3].).

Terms	Definition
Coherent Radar	Transmitted signal has a constant phase relationship to an oscillator in the transmitter.
Frequency-Agile Radar	Pulse or group of pulses is transmitted at different frequencies.
LPI Radar	A radar with parameters that make it difficult for an ES receiver to correctly identify the radar type
Quiet Range	The range where radar detects a target at the same time the target can detect the radar's signal.
Random–Signal Radar	A radar which uses a waveform that is truly random (e.g., noise)
Polyphase–coded Continuous-Wave Radar	A radar that has a pseudo-random phase–coded modulation on a transmitted continuous-wave signal

B. PRINCIPAL CONTRIBUTIONS

The objective of this research is primarily to examine the feasibility of classifying the LPI radar signals using image processing and neural networks and to extract the radar signal parameters after classification.

To obtain insight into the characteristics of LPI radar, a MATLAB simulation of polytime phase–coded signals was developed. The developed polytime phase–code waveform simulation is listed in Appendix A. The implementation was further incorporated with a MATLAB toolbox containing several LPI waveforms developed in previous thesis work [2, 3].

Using previous thesis work Refs. [3-6] for detecting the LPI radar signals, a classification process using image processing and neural networks was developed. The classification process uses the time–frequency and bifrequency (2-D) representations that are a result of the following detection methods:

- Cyclostationary processing—bifrequency
- Quadrature mirror filter bank—time–frequency
- Wigner–Ville distribution—time–frequency

The result of this part of the work is a signal processing architecture that provides a semi-automatic classification of a LPI signal. The code is enclosed in Appendix B.

Finally a method was examined to extract autonomously the LPI signal parameters from the time–frequency and bifrequency representations. This part of the work was a concept study resulting in suggestions of suitable methods to extract the parameters from the classified signals.

C. THESIS OUTLINE

The purpose of this thesis is to document (a) the development of a simulation of the polytime phase–coded signal, (b) the implementation of the classification process of LPI radar waveforms, and (c) the study of suitable parameter extraction methods. In order to do this the thesis is organized as follows:

Chapter II briefly describes the LPI radar waveforms used in this thesis work and their spectral properties. Emphasis lies on implementing polytime phase–coded signals, frequency modulated continuous wave signals (FMCW), binary phase shift keying signals (BPSK), phase–coded signals (P1-4) and Frank code signals (FR). More information about different LPI radar signal types can be found in [1-5].

Chapter III describes the detection methods developed in previous thesis work and their behavior. The focus for the chapter is on the output images and on the difference in images produced by the various detection methods for any given signal. For this chapter, references [2-5] provide a more detailed description.

Chapter IV begins with a general description of the proposed classification method. Next the pre–processing and feature extraction methods selected for this work are presented. The third part of the chapter describes the classification algorithms used in the work; the chapter ends with the results from classifying the test signals.

Chapter V briefly describes some testing performed to extract the significant parameters of a LPI radar signal once it has been classified.

Chapter VII concludes the thesis and recommends future work.

Appendix A contains the MATLAB M-files used for implementation of the polytime phase–coded signal T1(n) through T2(n).

Appendix B contains the MATLAB M-files for the classification process described in the thesis.

THIS PAGE INTENTIONALLY LEFT BLANK

II. LOW PROBABILITY OF INTERCEPT EMITTERS AND THEIR SPECTRAL PROPERTIES

This work is about classifying a detected LPI signal. That is, the signal has already been detected and the initial signal processing has been performed. This chapter introduces the signals that are of interest to this work. Both the power spectrum magnitude and the periodic ambiguity function (PAF) for each signal are presented.

The PAF is a tool useful to analyze the response of a matched receiver. The function uses N copies of the reference (transmitted signal) function to cross correlate the return CW signal (correlation receiver) and perform target detection. The PAF is similar to the ambiguity function often used to represent the magnitude of the matched receiver output for a coherent pulse train [1]. If the signal has period T and the reference signal, $u(t)$, is constructed from an integral N number of periods of the transmitted signal, the PAF is defined as

$$\left|\chi_{NT}(\tau,\nu)\right| = \left|\frac{1}{NT}\int_0^{NT} u(t-\tau)u^*(t)e^{j2\pi\nu t}\,dt\right| \tag{2.1}$$

where τ is the delay, assumed to be constant, and ν is the Doppler shift that gives the delay rate of change.

The PAF has symmetry properties along the delay axis and the Doppler shift axis. On the delay axis this symmetry is described by

$$\left|\chi_{NT}(nT,\nu)\right| = \left|\chi_{NT}(0,\nu)\right| \tag{2.2}$$

for any integer n. A periodicity of T is also maintained anywhere on the cuts $\nu = m/T$. With $m = 0, \pm 1, \pm 2, \dots$, the symmetry on the Doppler axis is described by

$$\left|\chi_{NT}(\tau, m/T)\right| = \left|\chi_{NT}(\tau + nT, m/T)\right|. \tag{2.3}$$

An example of the application of the PAF on a Frank–coded signal number of phase–codes, $N_c = 64$, number of cycles per phase, $cpp = 1$ and number of reference signals used in the correlation receiver, $N = 1$ is shown in Figure 1. The two axes describe the delay, τ, and the Doppler shift, ν, which are added to the received signal compared with what the receiver was matched to ($\tau = \nu = 0$). Both of these values are normalized in the plot, by the sub–code duration, t_b (delay axis), and by the code period, $N_c t_b$ (Doppler axis). The PAF repeats every integer multiple of the code period. From the example, $f_c = 1$ kHz, $f_s = 7$ kHz, $cpp = 1$, so on the delay axis the PAF repeats at $f_s N_c t_b = 448$ and on the Doppler axis at $1/T$.

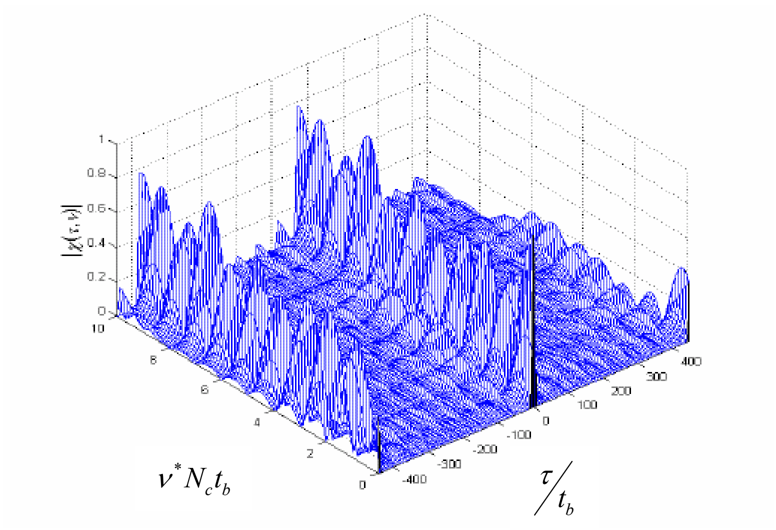


Figure 1 Illustration of the Periodic Ambiguity Function (PAF) Applied to a Frank–coded Signal (From [1].).

Reference [1] gives a detailed description of the PAF and References [1-5] gives a more in–depth description about signals not included in this work.

A. SELECTION OF SIGNAL TYPES

A reasonable limitation, which is made in this thesis work, is to look at one signal for classification from a typical group of signals. Care must be taken since it is important to test the classification algorithm on signals that have a similar time–frequency/bifrequency image representation. The Frank code and the P4 signal were selected as a good representation of polyphase coded signals. Only signals with a carrier frequency of 1 kHz were used. The sampling frequency is 7 kHz. Only a few variations of the signal parameters are used. Since this work focuses on concepts, a complete set of signals would be too time–consuming in combination with the rest of the work. Given these limitations, signals types and their variations are shown in Table 3. Table 4 and Table 5 show the signal variations used for testing the finished network. In Table 4 the signals to test variations in the modulation pattern are shown and in Table 5 the signals to test the noise variation are shown. A criterion, in this selection, was to only use noisy signals since a noise–less signal is not realistic.

Table 3 Signals for Training of Classifier. Common for all is $f_c = 1000$ Hz and $f_s = 7000$ Hz.

SIGNALS FOR TRAINING OF CLASSIFIER							
Signal	Barker bit length	Codes per Period, cpp	Modulation BW [Hz]	Modulation period, t_m [ms]	# of phase states	# of segments	Signal to Noise ratio, SNR [dB]
BPSK							
B_1_7_7_1_s	7	1	n/a	n/a	n/a	n/a	signal only
B_1_7_7_1_0	7	1	n/a	n/a	n/a	n/a	0
B_1_7_7_1_-3	7	1	n/a	n/a	n/a	n/a	-3
B_1_7_7_4_s	7	4	n/a	n/a	n/a	n/a	signal only
B_1_7_7_4_0	7	4	n/a	n/a	n/a	n/a	0
B_1_7_7_4_-3	7	4	n/a	n/a	n/a	n/a	-3
B_1_7_7_7_s	7	7	n/a	n/a	n/a	n/a	signal only
B_1_7_7_7_0	7	7	n/a	n/a	n/a	n/a	0
B_1_7_7_7_-3	7	7	n/a	n/a	n/a	n/a	-3
B_1_7_11_1_s	11	1	n/a	n/a	n/a	n/a	signal only
B_1_7_11_1_0	11	1	n/a	n/a	n/a	n/a	0
B_1_7_11_1_-3	11	1	n/a	n/a	n/a	n/a	-3
B_1_7_11_4_s	11	4	n/a	n/a	n/a	n/a	signal only
B_1_7_11_4_0	11	4	n/a	n/a	n/a	n/a	0
B_1_7_11_4_-3	11	4	n/a	n/a	n/a	n/a	-3
B_1_7_11_7_s	11	7	n/a	n/a	n/a	n/a	signal only
B_1_7_11_7_0	11	7	n/a	n/a	n/a	n/a	0
B_1_7_11_7_-3	11	7	n/a	n/a	n/a	n/a	-3
FMCW							
F_1_7_250_20_s	n/a	n/a	250	20	n/a	n/a	signal only
F_1_7_250_20_0	n/a	n/a	250	20	n/a	n/a	0
F_1_7_250_20_-3	n/a	n/a	250	20	n/a	n/a	-3
F_1_7_250_50_s	n/a	n/a	250	50	n/a	n/a	signal only
F_1_7_250_50_0	n/a	n/a	250	50	n/a	n/a	0
F_1_7_250_50_-3	n/a	n/a	250	50	n/a	n/a	-3
F_1_7_350_35_s	n/a	n/a	350	35	n/a	n/a	signal only
F_1_7_500_20_s	n/a	n/a	500	20	n/a	n/a	signal only
F_1_7_500_20_0	n/a	n/a	500	20	n/a	n/a	0
F_1_7_500_20_-3	n/a	n/a	500	20	n/a	n/a	-3
F_1_7_500_50_s	n/a	n/a	500	50	n/a	n/a	signal only
F_1_7_500_50_0	n/a	n/a	500	50	n/a	n/a	0
F_1_7_500_50_-3	n/a	n/a	500	50	n/a	n/a	-3
Frank code							
FR_1_7_4_1_s	4	1	n/a	n/a	n/a	n/a	signal only
FR_1_7_4_1_0	4	1	n/a	n/a	n/a	n/a	0
FR_1_7_4_1_-3	4	1	n/a	n/a	n/a	n/a	-3
FR_1_7_4_4_s	4	4	n/a	n/a	n/a	n/a	signal only
FR_1_7_4_4_0	4	4	n/a	n/a	n/a	n/a	0
FR_1_7_4_4_-3	4	4	n/a	n/a	n/a	n/a	-3
FR_1_7_4_7_s	4	7	n/a	n/a	n/a	n/a	signal only
FR_1_7_4_7_0	4	7	n/a	n/a	n/a	n/a	0
FR_1_7_4_7_-3	4	7	n/a	n/a	n/a	n/a	-3
Polyphase code							
P4_1_7_16_1_s	16	1	n/a	n/a	n/a	n/a	signal only
P4_1_7_16_1_0	16	1	n/a	n/a	n/a	n/a	0
P4_1_7_16_1_-3	16	1	n/a	n/a	n/a	n/a	-3
P4_1_7_16_4_s	16	4	n/a	n/a	n/a	n/a	signal only
P4_1_7_16_4_0	16	4	n/a	n/a	n/a	n/a	0
P4_1_7_16_4_-3	16	4	n/a	n/a	n/a	n/a	-3
P4_1_7_16_7_s	16	7	n/a	n/a	n/a	n/a	signal only
P4_1_7_16_7_0	16	7	n/a	n/a	n/a	n/a	0
P4_1_7_16_7_-3	16	7	n/a	n/a	n/a	n/a	-3
Polytime Phase code							
PT1_1_7_2_4_s	n/a	n/a	n/a	n/a	2	4	signal only
PT1_1_7_2_4_0	n/a	n/a	n/a	n/a	2	4	0
PT1_1_7_2_4_-3	n/a	n/a	n/a	n/a	2	4	-3
PT1_1_7_3_4_s	n/a	n/a	n/a	n/a	3	4	signal only
PT1_1_7_4_4_s	n/a	n/a	n/a	n/a	4	4	signal only
PT1_1_7_4_4_0	n/a	n/a	n/a	n/a	4	4	0
PT1_1_7_4_4_-3	n/a	n/a	n/a	n/a	4	4	-3

Table 4 Signals for Testing of Classifier with Variation in Modulation. Common for all is $f_c = 1000$ Hz and $f_s = 7000$ Hz.

SIGNALS FOR TESTING OF CLASSIFIER, Modulation variations							
Signal	Barker bit length	Codes per Period, cpp	Modulation BW [Hz]	Modulation period, t_m [ms]	# of phase states	# of segments	Signal to Noise ratio, SNR [dB]
BPSK							
B_1_7_7_2_0	7	2	n/a	n/a	n/a	n/a	0
B_1_7_7_3_0	7	3	n/a	n/a	n/a	n/a	0
B_1_7_7_5_0	7	5	n/a	n/a	n/a	n/a	0
B_1_7_7_6_0	7	6	n/a	n/a	n/a	n/a	0
B_1_7_11_2_0	11	2	n/a	n/a	n/a	n/a	0
B_1_7_11_3_0	11	3	n/a	n/a	n/a	n/a	0
B_1_7_11_5_0	11	5	n/a	n/a	n/a	n/a	0
B_1_7_11_6_0	11	6	n/a	n/a	n/a	n/a	0
FMCW							
F_1_7_350_35_0	n/a	n/a	350	35	n/a	n/a	0
F_1_7_350_35_-3	n/a	n/a	350	35	n/a	n/a	-3
Frank code							
FR_1_7_4_2_0	4	2	n/a	n/a	n/a	n/a	0
FR_1_7_4_3_0	4	3	n/a	n/a	n/a	n/a	0
FR_1_7_4_5_0	4	5	n/a	n/a	n/a	n/a	0
FR_1_7_4_6_0	4	6	n/a	n/a	n/a	n/a	0
Polyphase code							
P4_1_7_16_2	16	2	n/a	n/a	n/a	n/a	0
P4_1_7_16_3	16	3	n/a	n/a	n/a	n/a	0
P4_1_7_16_5	16	5	n/a	n/a	n/a	n/a	0
P4_1_7_16_6	16	6	n/a	n/a	n/a	n/a	0
Polytime Phase code							
PT1_1_7_3_4_0	n/a	n/a	n/a	n/a	3	4	0
PT1_1_7_3_4_-3	n/a	n/a	n/a	n/a	3	4	-3

Table 5 Signals for Testing of Classifier with Variation in Noise Level. Common for all is $f_c = 1000$ Hz and $f_s = 7000$ Hz.

SIGNALS FOR TESTING OF CLASSIFIER, Noise variations							
Signal	Barker bit length	Codes per Period, cpp	Modulation BW [Hz]	Modulation period, t_m [ms]	# of phase states	# of segments	Signal to Noise ratio, SNR [dB]
BPSK							
B_1_7_7_1_-2	7	1	n/a	n/a	n/a	n/a	-2
B_1_7_7_1_-1	7	1	n/a	n/a	n/a	n/a	-1
B_1_7_7_1_1	7	1	n/a	n/a	n/a	n/a	1
B_1_7_7_1_2	7	1	n/a	n/a	n/a	n/a	2
B_1_7_7_1_3	7	1	n/a	n/a	n/a	n/a	3
B_1_7_7_4_-2	7	4	n/a	n/a	n/a	n/a	-2
B_1_7_7_4_-1	7	4	n/a	n/a	n/a	n/a	-1
B_1_7_7_4_1	7	4	n/a	n/a	n/a	n/a	1
B_1_7_7_4_2	7	4	n/a	n/a	n/a	n/a	2
B_1_7_7_4_3	7	4	n/a	n/a	n/a	n/a	3
B_1_7_7_7_-2	7	7	n/a	n/a	n/a	n/a	-2
B_1_7_7_7_-1	7	7	n/a	n/a	n/a	n/a	-1
B_1_7_7_7_1	7	7	n/a	n/a	n/a	n/a	1
B_1_7_7_7_2	7	7	n/a	n/a	n/a	n/a	2
B_1_7_7_7_3	7	7	n/a	n/a	n/a	n/a	3
B_1_7_11_1_-2	11	1	n/a	n/a	n/a	n/a	-2
B_1_7_11_1_-1	11	1	n/a	n/a	n/a	n/a	-1
B_1_7_11_1_1	11	1	n/a	n/a	n/a	n/a	1
B_1_7_11_1_2	11	1	n/a	n/a	n/a	n/a	2
B_1_7_11_1_3	11	1	n/a	n/a	n/a	n/a	3
B_1_7_11_4_-2	11	4	n/a	n/a	n/a	n/a	-2
B_1_7_11_4_-1	11	4	n/a	n/a	n/a	n/a	-1
B_1_7_11_4_1	11	4	n/a	n/a	n/a	n/a	1
B_1_7_11_4_2	11	4	n/a	n/a	n/a	n/a	2
B_1_7_11_4_3	11	4	n/a	n/a	n/a	n/a	3
B_1_7_11_7_-2	11	7	n/a	n/a	n/a	n/a	-2
B_1_7_11_7_-1	11	7	n/a	n/a	n/a	n/a	-1
B_1_7_11_7_1	11	7	n/a	n/a	n/a	n/a	1
B_1_7_11_7_2	11	7	n/a	n/a	n/a	n/a	2
B_1_7_11_7_3	11	7	n/a	n/a	n/a	n/a	3
FMCW							
F_1_7_250_20_-2	n/a	n/a	250	20	n/a	n/a	-2
F_1_7_250_20_-1	n/a	n/a	250	20	n/a	n/a	-1
F_1_7_250_20_1	n/a	n/a	250	20	n/a	n/a	1
F_1_7_250_20_2	n/a	n/a	250	20	n/a	n/a	2
F_1_7_250_20_3	n/a	n/a	250	20	n/a	n/a	3
F_1_7_250_50_-2	n/a	n/a	250	50	n/a	n/a	-2
F_1_7_250_50_-1	n/a	n/a	250	50	n/a	n/a	-1
F_1_7_250_50_1	n/a	n/a	250	50	n/a	n/a	1
F_1_7_250_50_2	n/a	n/a	250	50	n/a	n/a	2
F_1_7_250_50_3	n/a	n/a	250	50	n/a	n/a	3
F_1_7_500_20_-2	n/a	n/a	500	20	n/a	n/a	-2
F_1_7_500_20_-1	n/a	n/a	500	20	n/a	n/a	-1
F_1_7_500_20_1	n/a	n/a	500	20	n/a	n/a	1
F_1_7_500_20_2	n/a	n/a	500	20	n/a	n/a	2
F_1_7_500_20_3	n/a	n/a	500	20	n/a	n/a	3
F_1_7_500_50_-2	n/a	n/a	500	50	n/a	n/a	-2
F_1_7_500_50_-1	n/a	n/a	500	50	n/a	n/a	-1
F_1_7_500_50_1	n/a	n/a	500	50	n/a	n/a	1
F_1_7_500_50_2	n/a	n/a	500	50	n/a	n/a	2
F_1_7_500_50_3	n/a	n/a	500	50	n/a	n/a	3

Frank code							
FR_1_7_4_1_-2	4	1	n/a	n/a	n/a	n/a	-2
FR_1_7_4_1_-1	4	1	n/a	n/a	n/a	n/a	-1
FR_1_7_4_1_1	4	1	n/a	n/a	n/a	n/a	1
FR_1_7_4_1_2	4	1	n/a	n/a	n/a	n/a	2
FR_1_7_4_1_3	4	1	n/a	n/a	n/a	n/a	3
FR_1_7_4_4_-2	4	4	n/a	n/a	n/a	n/a	-2
FR_1_7_4_4_-1	4	4	n/a	n/a	n/a	n/a	-1
FR_1_7_4_4_1	4	4	n/a	n/a	n/a	n/a	1
FR_1_7_4_4_2	4	4	n/a	n/a	n/a	n/a	2
FR_1_7_4_4_3	4	4	n/a	n/a	n/a	n/a	3
FR_1_7_4_7_-2	4	7	n/a	n/a	n/a	n/a	-2
FR_1_7_4_7_-1	4	7	n/a	n/a	n/a	n/a	-1
FR_1_7_4_7_1	4	7	n/a	n/a	n/a	n/a	1
FR_1_7_4_7_2	4	7	n/a	n/a	n/a	n/a	2
FR_1_7_4_7_3	4	7	n/a	n/a	n/a	n/a	3
Polyphase code							
P4_1_7_16_1_-2	16	1	n/a	n/a	n/a	n/a	-2
P4_1_7_16_1_-1	16	1	n/a	n/a	n/a	n/a	-1
P4_1_7_16_1_1	16	1	n/a	n/a	n/a	n/a	1
P4_1_7_16_1_2	16	1	n/a	n/a	n/a	n/a	2
P4_1_7_16_1_3	16	1	n/a	n/a	n/a	n/a	3
P4_1_7_16_4_-2	16	4	n/a	n/a	n/a	n/a	-2
P4_1_7_16_4_-1	16	4	n/a	n/a	n/a	n/a	-1
P4_1_7_16_4_1	16	4	n/a	n/a	n/a	n/a	1
P4_1_7_16_4_2	16	4	n/a	n/a	n/a	n/a	2
P4_1_7_16_4_3	16	4	n/a	n/a	n/a	n/a	3
P4_1_7_16_7_-2	16	7	n/a	n/a	n/a	n/a	-2
P4_1_7_16_7_-1	16	7	n/a	n/a	n/a	n/a	-1
P4_1_7_16_7_1	16	7	n/a	n/a	n/a	n/a	1
P4_1_7_16_7_2	16	7	n/a	n/a	n/a	n/a	2
P4_1_7_16_7_3	16	7	n/a	n/a	n/a	n/a	3
Polytime Phase code							
PT1_1_7_2_4_-2	n/a	n/a	n/a	n/a	3	4	signal only
PT1_1_7_2_4_-1	n/a	n/a	n/a	n/a	3	4	0
PT1_1_7_2_4_1	n/a	n/a	n/a	n/a	3	4	signal only
PT1_1_7_2_4_2	n/a	n/a	n/a	n/a	3	4	0
PT1_1_7_2_4_3	n/a	n/a	n/a	n/a	3	4	-3
PT1_1_7_4_4_-2	n/a	n/a	n/a	n/a	3	4	signal only
PT1_1_7_4_4_-1	n/a	n/a	n/a	n/a	3	4	0
PT1_1_7_4_4_1	n/a	n/a	n/a	n/a	3	4	signal only
PT1_1_7_4_4_2	n/a	n/a	n/a	n/a	3	4	0
PT1_1_7_4_4_3	n/a	n/a	n/a	n/a	3	4	-3

B. POLYTIME PHASE–CODES

As with the Frank, P1, P2, P3, and P4 polyphase codes [7], the polytime phase–code is developed by letting the phase change approximate a stepped–frequency or linear frequency modulation waveform. The difference is that the subcode period is not uniform in size. That is, in the previous signals, the size of the phase step varies as needed to approximate the underlying waveform and the time spent at any given phase state is a constant. In the polytime phase–code the approximation of a stepped–frequency or linear frequency modulation waveform is performed by quantization of the underlying waveform into a user–selected number of phase states. This results in the time spent on each phase state changing throughout the duration of the waveform [8].

Four types of polytime waveforms exist. The first two variants of polytime coded waveforms, denoted T1(n) and T2(n) where n is the number of phase states, can be generated using the stepped frequency model to approximate the underlying waveform. The T3(n) and T4(n) polytime waveforms are approximations of a linear frequency modulation model. Increasing the number of phase states increases the quality of the polytime approximation to the underlying waveform, but it also reduces the time spent at any given phase state and, therefore, complicates the generation of the waveform. [8] As part of this work, a program to generate the four different polytime phase–codes was created. The code is included in Appendix A.

1. Polytime Code T1

The T1(n) sequence is generated using the stepped–frequency waveform that is "zero–beat" at the leading segment, meaning that the first code segment is at "zero" frequency. The equation for the wrapped phase, $\varphi(t)$, versus time for the T1(n) polytime sequence is [7]

$$\varphi(t) = \mathrm{MOD}\left\{\frac{2\pi}{n}\mathrm{INT}\left[(kt - jT)\frac{jn}{T}\right], 2\pi\right\} \tag{1.4}$$

where $j = 0,1,2,\ldots,k-1$ is the segment number in the stepped RF waveform, k is the number of segments in the T1 code sequence, t is time, T is the overall code duration, and n is the number of phase states in the code sequence.

An example of the relationship between a stepped RF waveform and its conversion into a T1(2) polytime waveform with $k = 4$ segments and $n = 2$ phase steps is shown in Figure 2. The figure shows how the polytime code phase steps are derived to fit the ideal RF phase. In this case two phase states are used (each phase step is π radians) and, as seen in the figure, the time between the two distinct phase steps is shortened to fit the derived phase to the ideal phase.

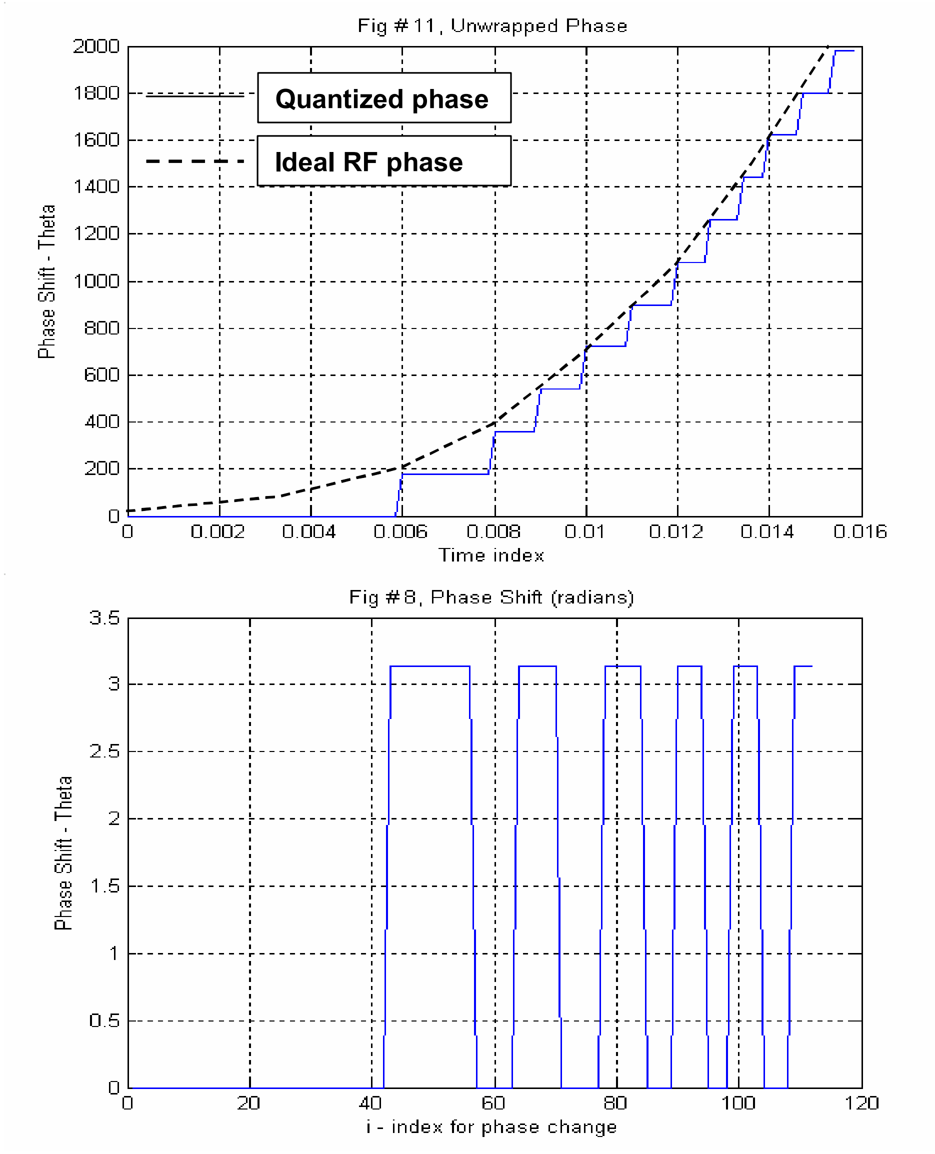


Figure 2 Polytime Waveform Derived from Linear FM Waveform.

The power spectrum magnitude of a T1(2) signal with $f_c = 1$ kHz, $n = 2$, $T = 0.016$ s, $N_c = 64$ and $N = 1$ is shown in Figure 3. The plot illustrates the wide bandwidth of the signal. Figure 4 is a plot of the PAF for the same T1(2) signal. This plot shows the relatively high sidelobes that the signal has in the Doppler plane.

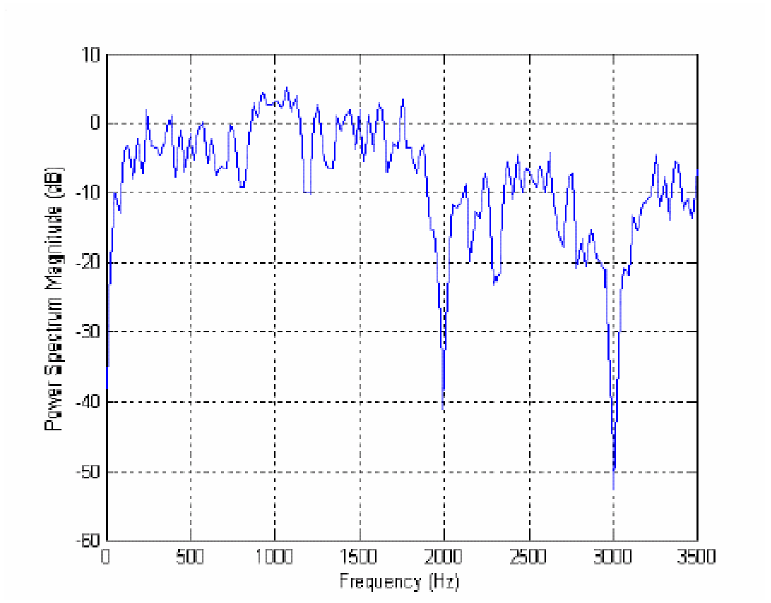


Figure 3 Polytime Code T1(2) Power Spectrum Magnitude (From [1].).

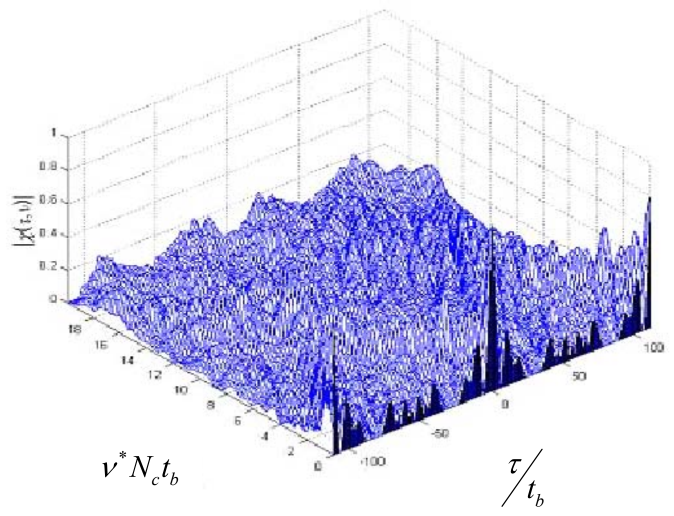


Figure 4 T1(2) Code PAF (From [1].).

2. Polytime Code T2

The T2(n) sequence is generated by approximating a stepped–frequency waveform that is zero–beat at the center frequency. If the waveform has an odd number of segments, the zero–beat frequency is the frequency of the center segment. If an even number of segments are used, the zero frequency is the frequency halfway between the two center most segments. The expression for the wrapped phase versus time for the T2(n) polytime sequence is [8]

$$\varphi(t) = \mathrm{MOD}\left\{\frac{2\pi}{n}\mathrm{INT}\left[(kt - jT)\left(\frac{2j-k+1}{T}\right)\frac{n}{2}\right], 2\pi\right\} \tag{1.5}$$

where the variables are the same as defined under T1(n).

Illustrated in Figure 5 is the power spectrum magnitude of a T2(2) signal with $f_c = 1\text{ kHz}$, $n = 2$, $T = 0.016\text{ s}$, $N_c = 64$ and $N = 1$. The peak side lobe level is approximately the same as the T1(2) examined above. One thing to notice is the dip that occurs around the center frequency. Figure 6 shows the PAF and, as with T1(2), the T2(2) also shows fairly large sidelobes.

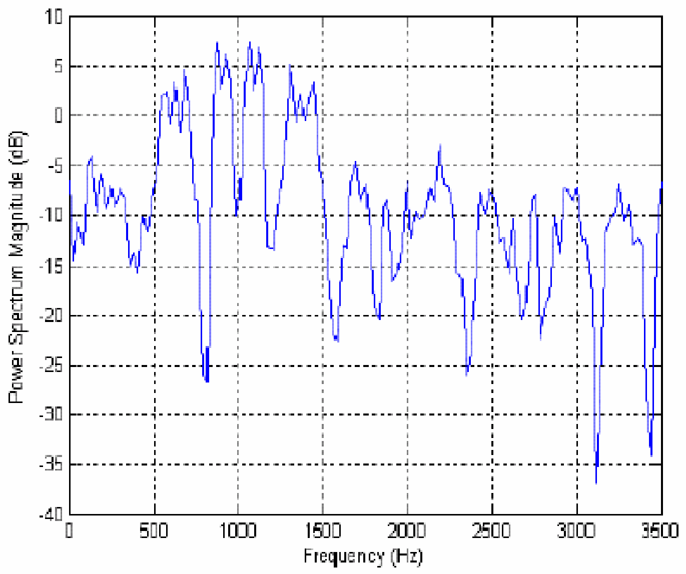


Figure 5 Polytime Code T2(2) Power Spectrum Magnitude (From [1].).

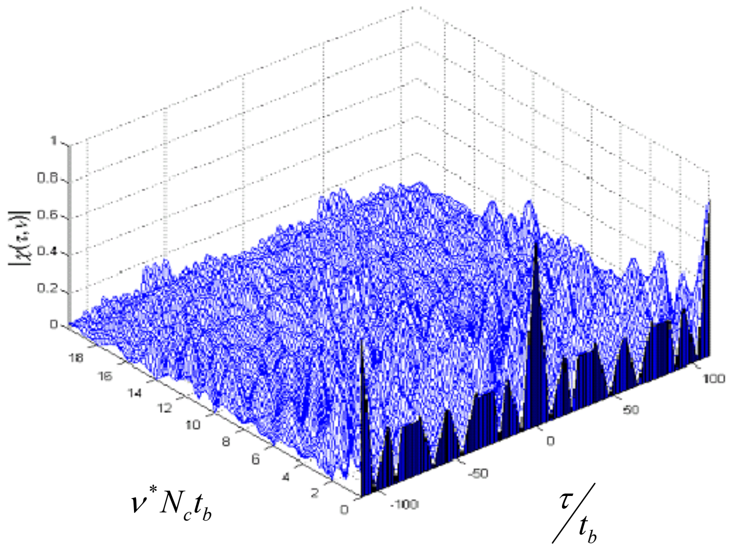


Figure 6 T2(2) Code PAF (From [1].).

3. Polytime Code T3

The T3 code has a linear FM underlying waveform. The T3(n) is zero beat at its leading edge. The equation for the wrapped phase versus time for a T3 polytime sequence is [8]

$$\varphi(t) = \text{MOD}\left\{\frac{2\pi}{n}\text{INT}\left[\frac{n\Delta F t^2}{2T}\right], 2\pi\right\} \tag{1.6}$$

where t is the time, T is the overall pulse duration, ΔF is the modulation bandwidth and n is the number of phase states in the code sequence. The power spectrum magnitude of the T3(2) signal with $f_c = 1$ kHz, $n = 2$, $T = 0.016$ s, $\Delta f = 1$ kHz, $N_c = 64$ and $N = 1$ is shown in Figure 7.

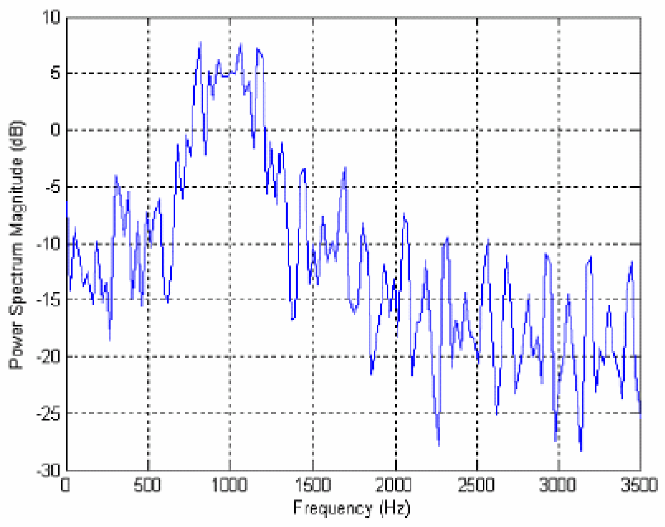


Figure 7 Polytime Code T3(2) Power Spectrum Magnitude (From [1].).

Figure 8 shows the PAF for the same T3(2) code as in Figure 7. As with previous polytime coded signals, the sidelobes are relatively high.

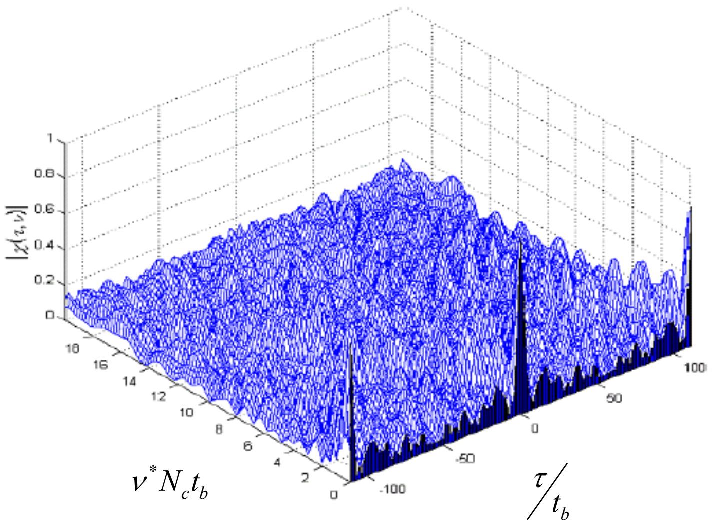


Figure 8 T3(2) Code PAF (From [1].).

4. Polytime Code T4

Polytime signal T4(n) also has an underlying linear FM waveform to generate the signal. Compared with T3(n), it has its zero–beat at the center frequency. The equation for the wrapped phase versus time for a T4(n) polytime sequence is [8]

$$\varphi(t) = \mathrm{MOD}\left\{\frac{2\pi}{n}\mathrm{INT}\left[\frac{n\Delta F t^2}{2T} - \frac{n\Delta F t}{2}\right], 2\pi\right\}. \tag{1.7}$$

The variables are the same as defined under T3(n).

The power spectrum magnitude of the T4(2) signal with $f_c = 1\text{ kHz}$, $n = 2$, $T = 0.016\text{ s}$, $\Delta f = 1\text{ kHz}$, $N_c = 64$ and $N = 1$ is shown in Figure 9. In this plot, the dip around the center frequency is obvious. Figure 10 shows the PAF for the T4(2) signal and shows similar performance as the T1(2) through T3(2). One conclusion is that the sidelobe levels are slightly bigger for the waveforms with zero–beat at the center frequency.

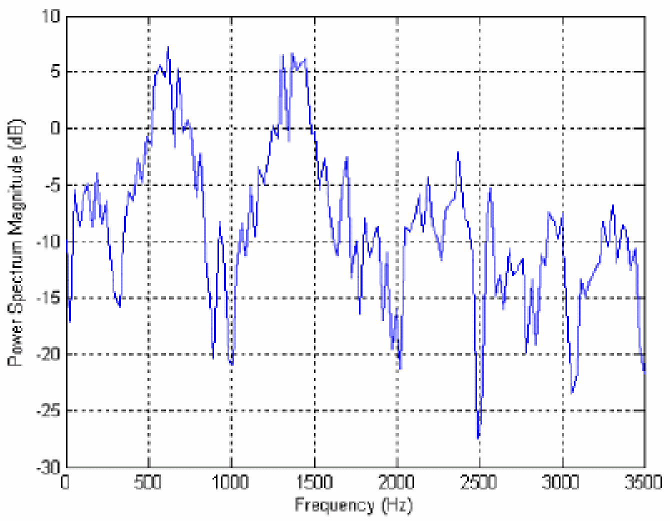


Figure 9 Polytime Code T4(2) Power Spectrum Magnitude (From [1].).

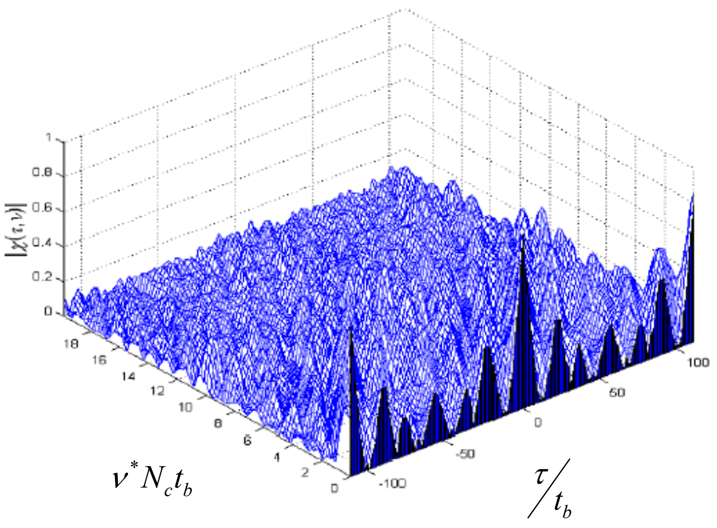


Figure 10 T4(2) Code PAF (From [1].).

C. BINARY PHASE SHIFT KEYING

Binary Phase Shift–Keying (BPSK) is a modulation technique that normally is not used in LPI applications; nonetheless, the modulation technique has proven to be extremely effective in communication and radar systems. As such, BPSK is a good test modulation to evaluate the proposed method of classification.

The signal $x(t)$ is a continuous wave (CW) sinusoid. After sampling the signal at $f_s = 7$ kHz, the modulated signal is created by adding a Barker code with length N_c. The property significant for this modulation type is low sidelobes at zero Doppler. As a result, this modulation technique is widely used. [3]

Figure 11(a) shows the sampled signal and modulating signal and Figure 11(b) presents the modulated signal for a 13-bit Barker code. The dashed red lines in (a) represent the modulating signal waveform. The number of periods of the carrier frequency per Barker Bit is equal to one ($cpp = 1$), meaning that one full period of the sampled signal fits within one bit of the 13-bit Barker code. The first five bits of the Barker code are $+1$ and the next 2 bits are -1, the result is five full periods under the first $+1$ portion of the modulating waveform, 2 full periods under the -1 portion of the modulating waveform, and so forth. In this figure, a complete 13-bit Barker code is represented.

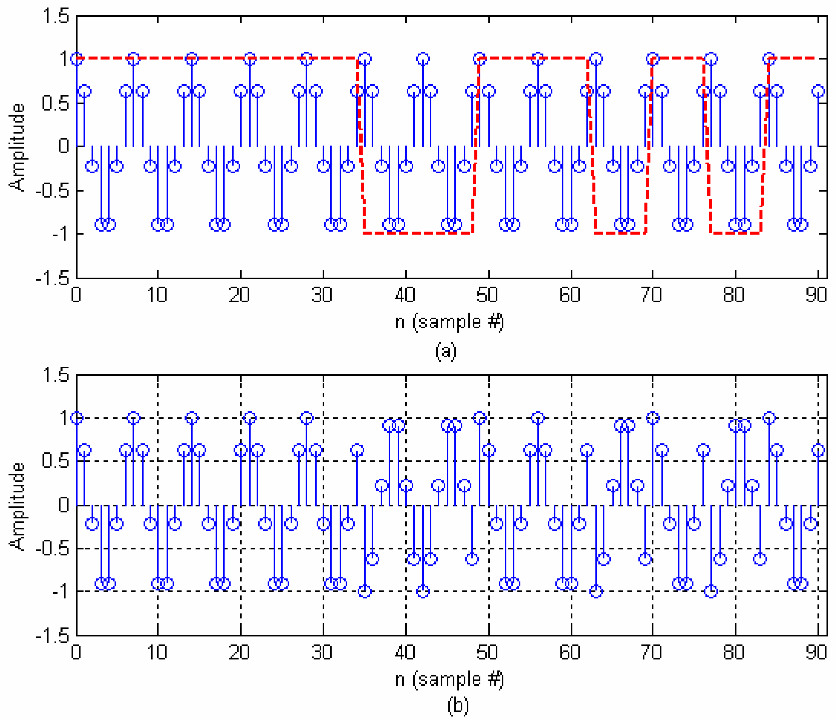


Figure 11 (a) Sampled Signal and Modulating Signal in Red (b) Modulated Signal for a 13-bit Barker Code BPSK Signal (From [3].).

In Table 6, the Barker code sequences for 7, 11 and 13 bits are shown together with the corresponding sidelobe levels.

Table 6 Barker Code Sequences (From [5].).

Code Length	Code Elements	Peak Sidelobe Level, dB
2	+−, ++	−6.0
3	++−	−9.5
4	++−+, +++−	−12.0
5	+++−+	−14.0
7	+++−−+−	−16.9
11	+++−−−+−−+−	−20.8
13	+++++−−++−+−+	−22.3

The power spectrum magnitude of a 7-bit BPSK signal with carrier frequency equal to 1000 Hz, the sampling frequency equal to 7000 Hz and 1 cycle per bit is shown in Figure 12.

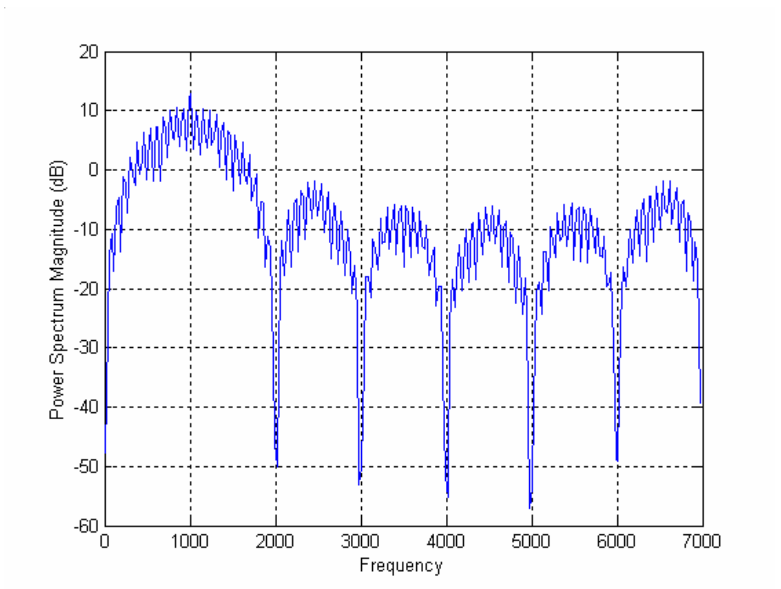


Figure 12 Power Spectrum Magnitude of a BPSK Signal Modulated with 7-bit Barker Code (From [1].).

Figure 13 shows the contour plot of the Periodic Ambiguity Function (PAF) of the 13-bit Barker CW signal. This plot describes the response of the correlation receiver to the signal modulated by the 13-bit Barker code. The response is a function of both the delay and Doppler. The figure shows the relatively large Doppler sidelobes this modulation has. Also note the constant sidelobe level at zero Doppler.

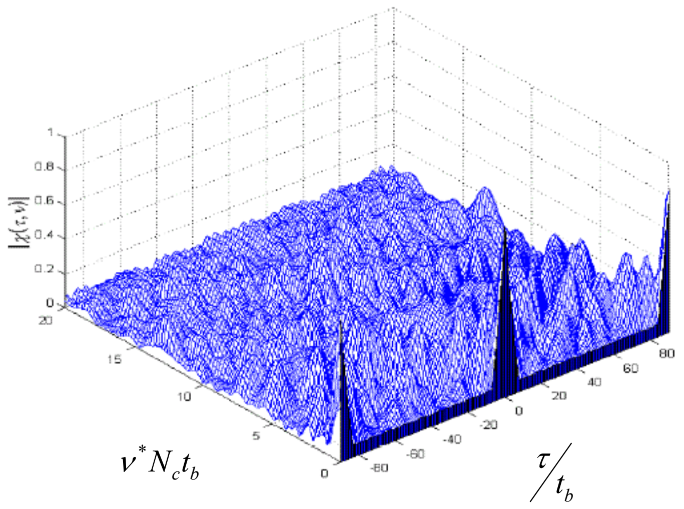
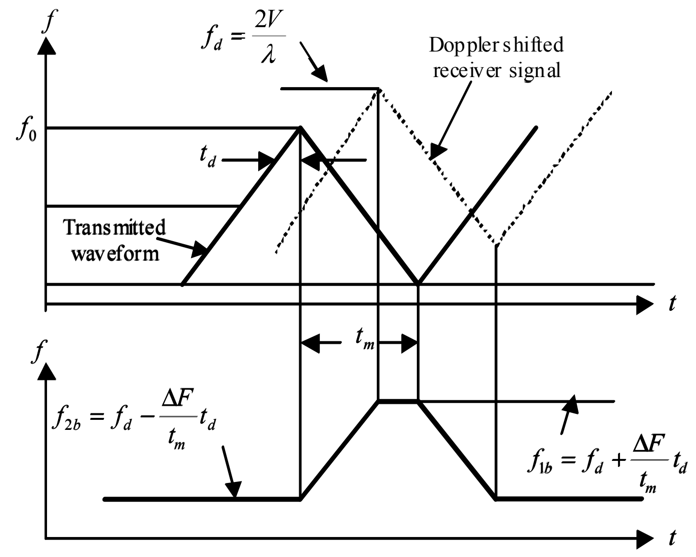


Figure 13 PAF for 13-bit Barker Binary PSK Signal Showing the large Doppler Sidelobes (From [1].).

D. FREQUENCY MODULATED CONTINUOUS WAVE

One of the most popular LPI waveforms is the triangular modulated frequency modulated continuous waveform (FMCW). The duty–cycle of the linear FMCW emitter is 100 % since it is transmitted continuously. Both the target range and the Doppler information can be measured unambiguously. With a low power output it maintains a low probability of intercept. The continuous output also results in that the FMCW waveform representing the best use of the output power available from solid–state devices. [1]

The triangular FMCW and the Doppler–shifted signal are shown in Figure 14. This is "built" by the use of two linear frequency modulation sections with positive and negative slopes. By mixing the Doppler–shifted received signal with the transmitted signal, the target beat frequencies can be obtained as shown. By taking the sum and the difference of the two beat frequencies, unambiguously extracting the range and Doppler frequency of the detected target is possible [1].



f
$f_d = \frac{2V}{\lambda}$
Doppler shifted receiver signal
f_0
t_d
Transmitted waveform
t
t_m
f
$f_{2b} = f_d - \frac{\Delta F}{t_m} t_d$
$f_{1b} = f_d + \frac{\Delta F}{t_m} t_d$
t

Figure 14 Linear Frequency Modulated Triangular Waveform and the Doppler–Shifted Signal (From [6].).

To illustrate the power spectrum magnitude of the signal an example of a FMCW signal with a modulation period of 20 ms. with carrier frequency at 1000 Hz and modulation BW of 500 Hz is shown in Figure 15.

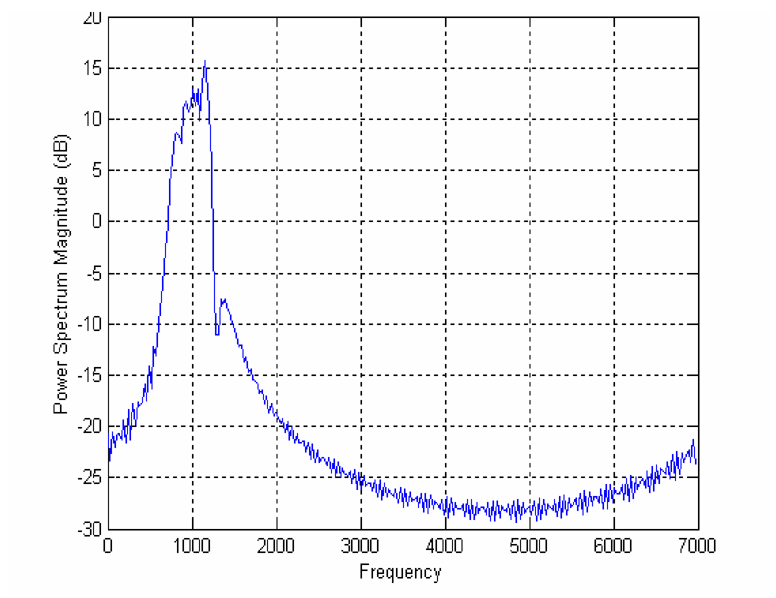


Figure 15 Power Spectrum Magnitude for an FMCW Signal (From [1].).

To end this section the PAF of the triangular FMCW signal is illustrated in Figure 16. The triangular shape is clearly visible.

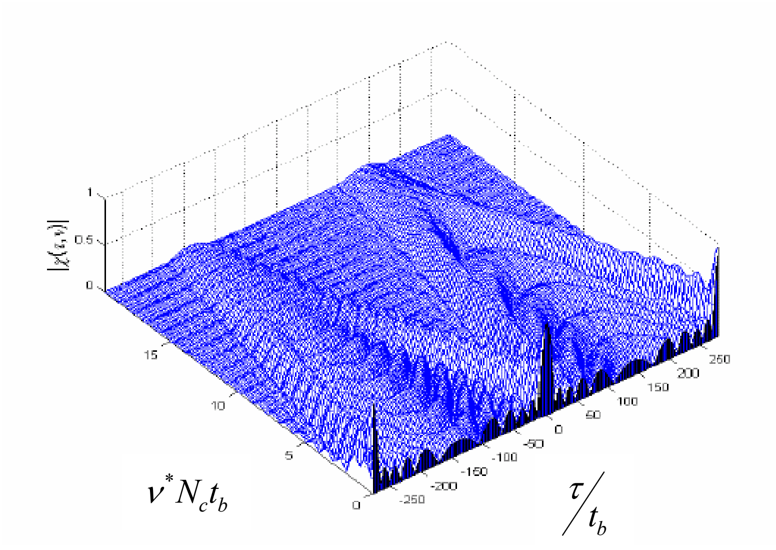


Figure 16 The PAF of a Triangular FMCW (From [1].).

E. POLYPHASE CODES

Polyphase codes have many useful features, such as low range–time sidelobes, ease of implementation, compatibility with digital implementation and low cross–correlation between codes. Polyphase codes also have compatibility with bandpass limited receivers. The highest obtainable pulse compression ratio that can be obtained with the Barker codes is 13, but with the Polyphase codes, code lengths of any size are possible [6].

The polyphase code waveforms provide a class of frequency derived phase–coded waveforms that can be sampled upon reception and processed digitally. Radar that use polyphase codes attempt to reduce interception from Signal Intelligence (SIGINT) receivers while being less likely to come under electronic attack (EA) and may prove to be the LPI waveform most commonly used in future applications [6]. This section describes the Frank–coded signal and the P4 polyphase coded signal. More information about the many other phase–coded signals can be found in Refs. [1-5].

1. Frank Code

The Frank code is one of the modulation codes that have been successfully implemented in LPI radars [1]. A Frank–coded waveform consists of a constant amplitude signal that is phase modulated by the phases of the Frank code.

The Frank code is derived from a step–approximation to a linear frequency modulation waveform using M frequency steps and M samples per frequency. The Frank code has a code–length or processing gain of $N_c = M^2$. The phase values of a Frank–coded signal, are given by the following equation:

$$\phi_{i,j} = \frac{2\pi}{M}(i-1)(j-1), \quad i = 1,2,\ldots,M \quad j = 1,2,\ldots,M \tag{1.8}$$

where $\phi_{i,j}$ describes the phase of the *i*-th sample of the *j*-th frequency.

Figure 17 illustrates the power spectrum magnitude of the Frank–coded signal with $M = 8$, $(N_c = 64)$. One cycle per phase is used to generate this signal. Some signal values can be identified in the plot, such as carrier frequency equal to 1000 Hz and the bandwidth of approximately 1000 Hz.

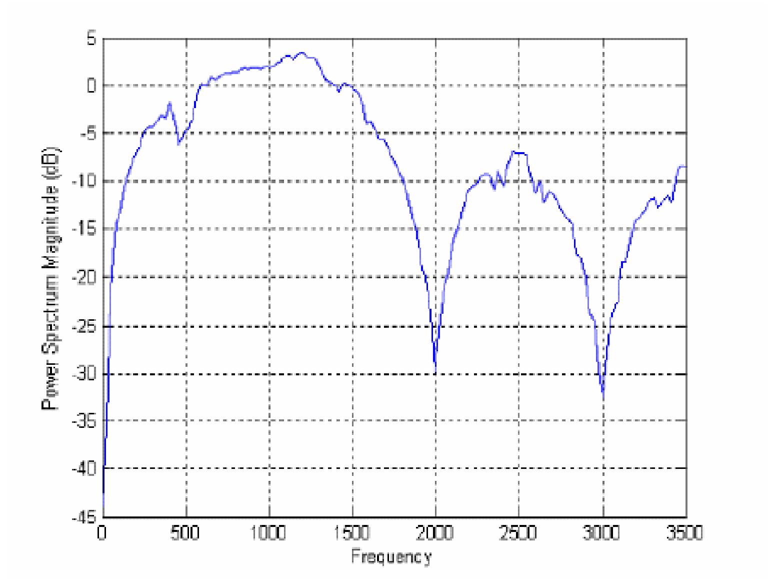


Figure 17 Power Spectrum Magnitude for a Frank–Coded Signal with *N*=8 (From [1].).

Figure 18 shows the PAF of the Frank–coded signal. The low sidelobe levels and the symmetry of the image are noticeable.

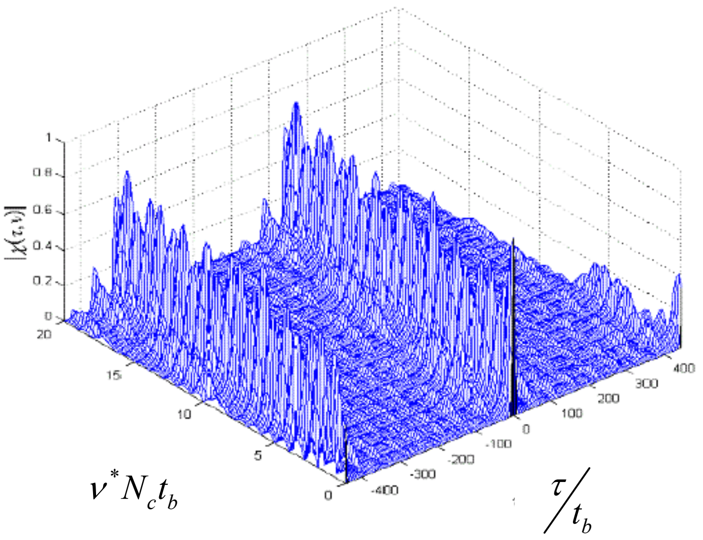


Figure 18 Frank Code PAF for $N_c = 64$, $N = 1$ (From [1].).

2. P4 CODE

The P4 code is derived by converting a linear frequency modulation waveform to baseband using a local oscillator on one end of the frequency sweep and sampling the *I* and *Q* video at the Nyquist rate. If it is assumed that the waveform has a pulse length *T,* in frequency $f = f_o + kT/2$, where *k* is a constant. The bandwidth *B* of the signal will be approximately $B = kT$ [4]. With this frequency, the phases of successive samples taken t_c apart are

$$\phi_i = 2\pi \int_0^{(i-1)t_c} \left[(f_0 + kt) - f_0 + \frac{kT}{2} \right] dt \tag{1.9}$$

or

$$\phi_i = 2\pi \int_0^{(i-1)t_c} k(t - T/2)\, dt \tag{1.10}$$

or

$$\phi_i = \pi k (i-1)^2 t_c^2 - \pi k T (i-1) t_c = \left[\frac{\pi (i-1)^2}{N_c} \right] - \pi (i-1) . \tag{1.11}$$

Figure 19 shows the power spectrum magnitude of the P4 signal with a carrier frequency of $f_c = 1$ kHz, 1 cycle per phase and bandwidth, $B = 1$ kHz.

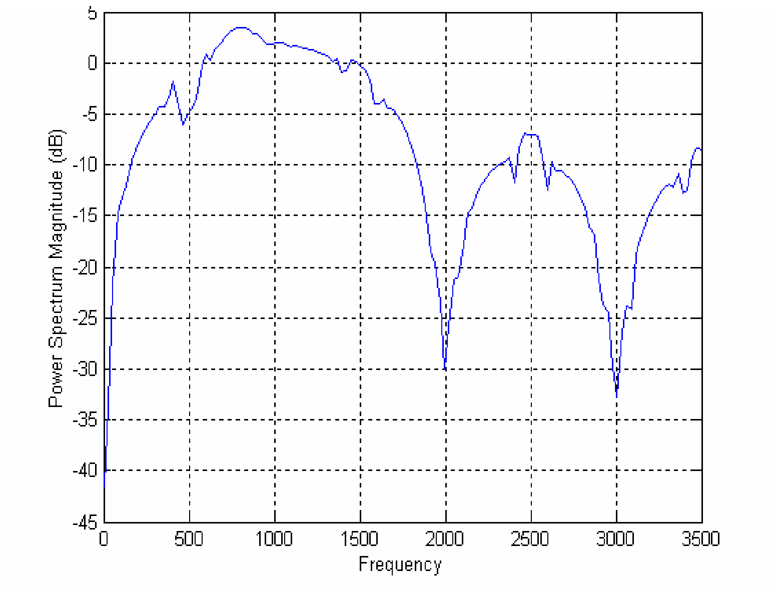


Figure 19 Power Spectrum Magnitude for a P4–Coded signal with $N_c = 64$ (From [1].).

Figure 20 shows the PAF of the P4–coded signal. This image shows that PAF for P4 is similar to the PAF for the Frank-code except that the peak sidelobes are slightly higher.

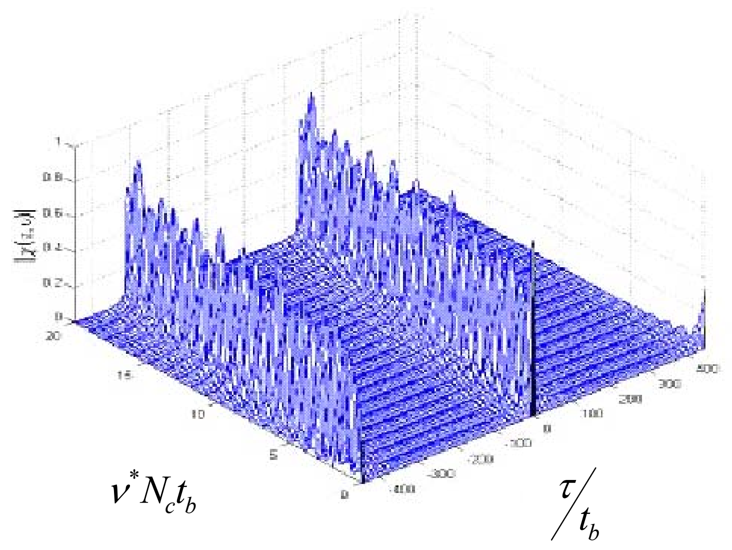


Figure 20 P4 code PAF for $N_c = 64$, $N = 1$ (From [1].).

F. SUMMARY

The signals to be used in the classification study are the T1(2), Frank, P4, BPSK and the FMCW. The description of the signals generally shows that problems might occur in a classification process due to their similarity and the fact that they are all derived from a linear FM waveform.

The next chapter will briefly discuss detection methods developed in earlier thesis work. These methods will now be used to extract the signals described in this chapter.

THIS PAGE INTENTIONALLY LEFT BLANK

III. DETECTION METHODS

This chapter briefly describes the detection methods used to generate the images used as input in the classification process. The methods have been developed in previous thesis work by Refs. [2-5] and have not been altered in this work except to remove the gridlines from the images.

A. CYCLOSTATIONARY PROCESSING

The implementation of cyclostationary theory for signal processing used in this thesis work was made by Refs. [5, 6]. The theory involves three main properties:

- Generation of spectral lines by quadratically transforming a signal;
- The statistical property called "second–order cyclostationarity," namely the periodic fluctuation of the auto–correlation function with time; and
- The correlation property for signal components in distinct spectral bands.

These properties form the link between the signal and its cyclostationary representation. The spectral lines are generated from the signal by putting the signal through a quadratic non–linear transformation. Components residing in the different spectral bands show random fluctuations known as second-order cyclostationarity. When the bands are correlated with each other using the autocorrelation function, a bifrequency representation of the signal is the resulting output.

The correlation integral is defined as

$$h(x) = \int_{-\infty}^{\infty} f(u)\, g(x+u)\; du \,. \tag{1.1}$$

Applying a fast Fourier transform (FFT), a Fourier transform pair is formed

$$\Im\big(h(x)\big) = F(s)\,G^{*}(s)\,. \tag{1.2}$$

If $f(x)$ and $g(x)$ are the same function, the integral above is normally called the *auto-correlation* function. It is the *cross–correlation* if they differ. The autocorrelation function is a quadratic transformation of a signal and may be interpreted as a measure of the predictability of the signal at time $t+\tau$ based on knowledge of the signal at time t. [4]

Looking at a time series of length T, the autocorrelation function is given by the time–average autocorrelation function:

$$R_x(T) \triangleq \lim_{T\to\infty} \frac{1}{T} \int_{-T/2}^{T/2} x\left(t+\frac{\tau}{2}\right) x^*\left(t-\frac{\tau}{2}\right) dt. \tag{1.3}$$

The cyclic autocorrelation function

$$R_x^\alpha(T) \triangleq \lim_{T\to\infty} \frac{1}{T} \int_{-T/2}^{T/2} x\left(t+\frac{\tau}{2}\right) x^*\left(t-\frac{\tau}{2}\right) e^{-j2\pi\alpha t}\, dt \tag{1.4}$$

shows the second–order periodicity of a time series $x(t)$ where α is the cycle frequency. The derivation of Equation (1.4) from (1.3) makes it possible to extract more information than if the autocorrelation function alone is only used. [4]

Due to the fact that the power spectrum magnitude may be obtained from the Fourier transform of the autocorrelation function, the spectral–correlation density (SCD) or the cyclic–spectral density may also be obtained from the Fourier transform of the cyclic autocorrelation function (1.4) [4]

$$S_x^\alpha(f) \triangleq \int_{-\infty}^{\infty} R_x^\alpha(\tau) e^{-j2\pi T}\, d\tau = \lim_{T\to\infty} \frac{1}{T} X_T\left(f+\frac{\alpha}{2}\right) X_T^*\left(f-\frac{\alpha}{2}\right) \tag{1.5}$$

where α is the cycle frequency and

$$X_T(f) \triangleq \int_{-T/2}^{T/2} x(u) e^{-j2\pi fu}\, du \tag{1.6}$$

which is the Fourier transform of the time domain signal $x(u)$. The additional variable α leads to a two–dimensional representation $S_x^\alpha(f)$ which is the bifrequency plane or (f,α) plane. [4]

In practice, the cyclic-spectral density must be estimated because the signals being processed are defined over a finite time interval (Δt). Estimates of the cyclic–spectral density can be obtained via time-smoothing or frequency–smoothing techniques. An estimate of the SCD using the time–smoothed cyclic periodogram is given by

$$S_x^\alpha(f) \approx S_{x_{T_W}}^\alpha(t,f)_{\Delta t} = \frac{1}{\Delta t}\int_{t-\Delta t/2}^{t+\Delta t/2} S_{x_{T_W}}(u,f)\, du, \tag{1.7}$$

where

$$S_{x_{T_W}}(u,f) = \frac{1}{T_W} X_{T_W}\left(u, f+\frac{\alpha}{2}\right) X_{T_W}^*\left(u, f-\frac{\alpha}{2}\right), \tag{1.8}$$

and Δt is the total observation time of the signal, T_W is the short–time FFT window length, and

$$X_{T_W}(u,f) = \int_{t-T_W/2}^{t+T_W/2} x(u) e^{-j2\pi fu}\, du \tag{1.9}$$

is the sliding short–time Fourier transform. Figure 21 shows that, for any signal $x(t)$, the frequency components are evaluated over a small time window T_W along the entire observation time interval Δt. The spectral components generated by each short–time Fourier Transform have a resolution, $\Delta f = 1/T_W$. The variable L is the overlapping factor between each short-time FFT. In order to avoid aliasing and cycle leakage on the estimates, the value of L is defined as $L \le T_W/4$. [4]

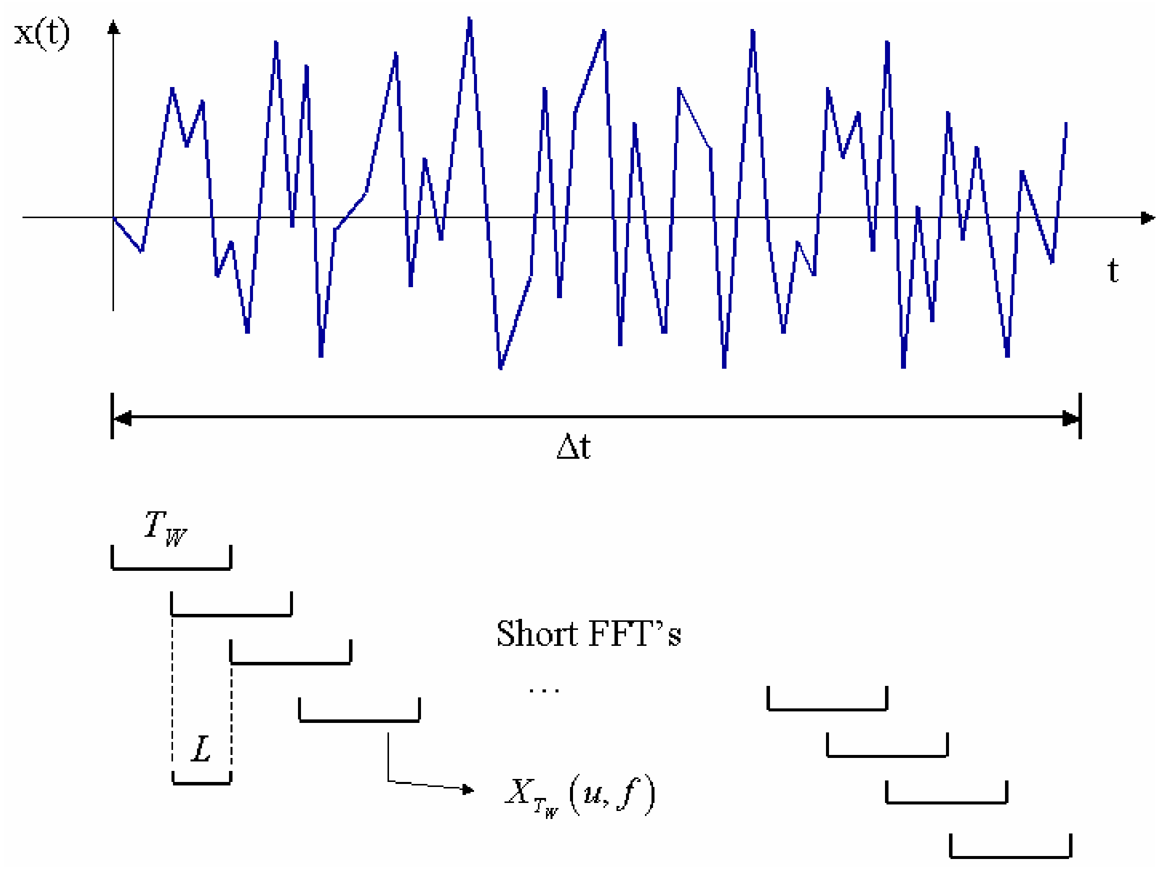


Figure 21 Pictorial Illustration of the Estimation of the Time–Variant Spectral Periodogram (From [4].).

Figure 22 shows that the spectral components of each short–time FFT are multiplied for the cyclic–spectrum estimates. Note that the dummy variable u has been replaced by the time instances $t_1, \ldots, t_p$. At each window (T_W), two components centered on some frequency f_0 and separated by some α_0 are multiplied together and the resulting sequence of products is then integrated over the total time (Δt).

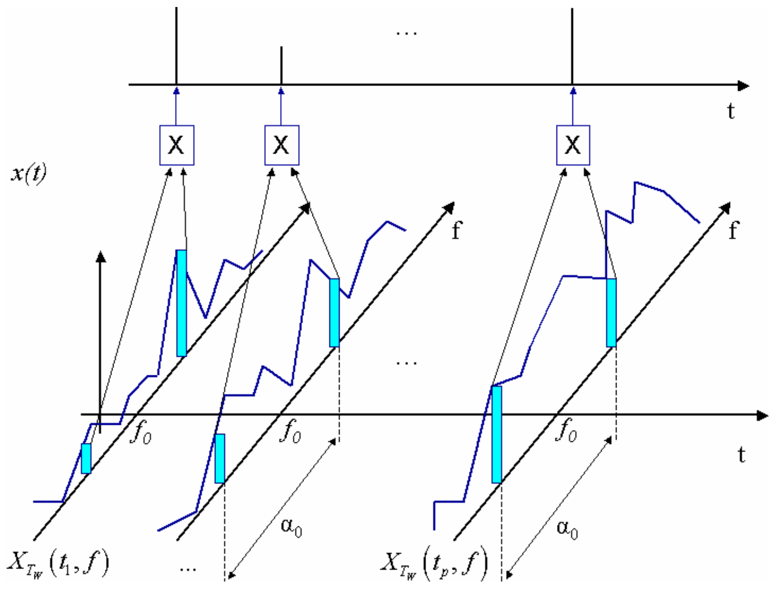


Figure 22 Sequence of Frequency Products for each Short–Time Fourier Transforms (From [4].).

The estimation $S_x^\alpha(f) \approx S_{x_{TW}}^\alpha(t, f)_{\Delta t}$ can be made as reliable and accurate as desired for any given t and Δf and, for all f, by making Δt sufficiently large. [4] Finally an illustration of the relationship between the frequency plane and the bifrequency plane is shown in Figure 23.

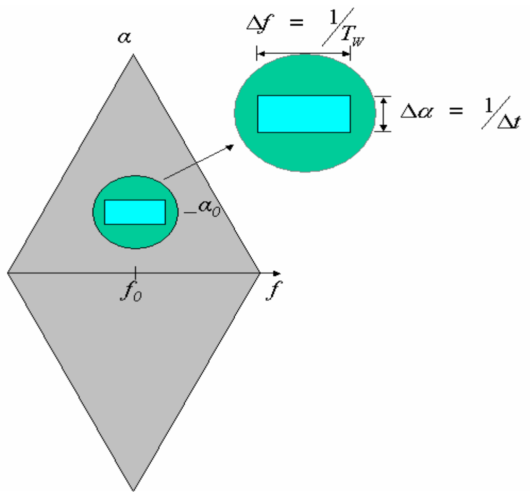


Figure 23 Bifrequency Plane, Frequency and Cycle Frequency Resolutions on Detailed Area (From [4].).

B. QUADRATURE MIRROR FILTER BANK

Complex sinusoids are used by the Fourier transform to perform the analysis of signals using appropriate basis functions. This approach is difficult since local information, such as an abrupt change in the signal, is spread out over all frequencies based on the infinite extension of the Fourier transform. This problem has been addressed by introducing windowed complex sinusoids as basis functions. This leads to the doubly indexed windowed Fourier transform:

$$X_{WF}(\omega,\tau)=\int_{-\infty}^{\infty} e^{-j\omega t} w(t-\tau)x(t)dt\,, \tag{1.10}$$

where $w(t-\tau)$ constitutes an appropriate window, $X_{WF}(\omega,\tau)$ is the Fourier transform of $x(t)$ windowed with $w(\cdot)$ and shifted by τ. An advantage of the windowed or short-time Fourier transform (STFT) is that, if a signal has most of its energy in a given time interval $\left[-T,T\right]$ and a given frequency interval $\left[-\Omega,\Omega\right]$, the STFT will be localized in the region $\left[-T,T\right]\times\left[-\Omega,\Omega\right]$ and will be close to zero in time and frequency intervals where the signal has little energy [6]. A negative aspect of the STFT is that a single window is used for all frequencies, meaning that the resolution of the analysis is the same at all locations in the time–frequency plane. Therefore arbitrarily high resolution in both time and frequency is not possible. [6]

By varying the window used, resolution in time can be traded for resolution in frequency. To isolate discontinuities in signals, it is possible to use some basis functions, which are very short, while longer ones are required to obtain a fine frequency analysis. The wavelet transform achieves this by obtaining the basis functions from a single prototype wavelet, $h_{ab}\left(t\right)$, with the use of translation and dilation/contraction as in

$$h_{a.b}(t)=\frac{1}{\sqrt{a}}h\left(\frac{t-b}{a}\right), \tag{1.11}$$

where a is a positive real number and b is a real number. For large a, the basis function becomes a stretched version of the prototype wavelet (low frequency function) while for small a, the basis function becomes a contracted wavelet (short high frequency function). The wavelet transform (WT) is defined as

$$X_W(a,b) = \frac{1}{\sqrt{a}} \int_{-\infty}^{\infty} h^*\left(\frac{t-b}{a}\right) x(t)dt. \qquad (1.12)$$

The time–frequency resolution of the WT involves a tradeoff not applicable to the STFT. At high frequencies, the WT is sharper in time, while at low frequencies, the WT is sharper in frequency [6].

These orthogonal wavelets can be implemented using quadrature mirror filters, which are filter pairs designed to divide the input signal energy into two orthogonal components based on the frequency. The basis function becomes a contracted wavelet, or a short high frequency function as it is in Figure 24 (a), and the wavelet transform divides the time–frequency plane into tiles as shown in Figure 24 (b).

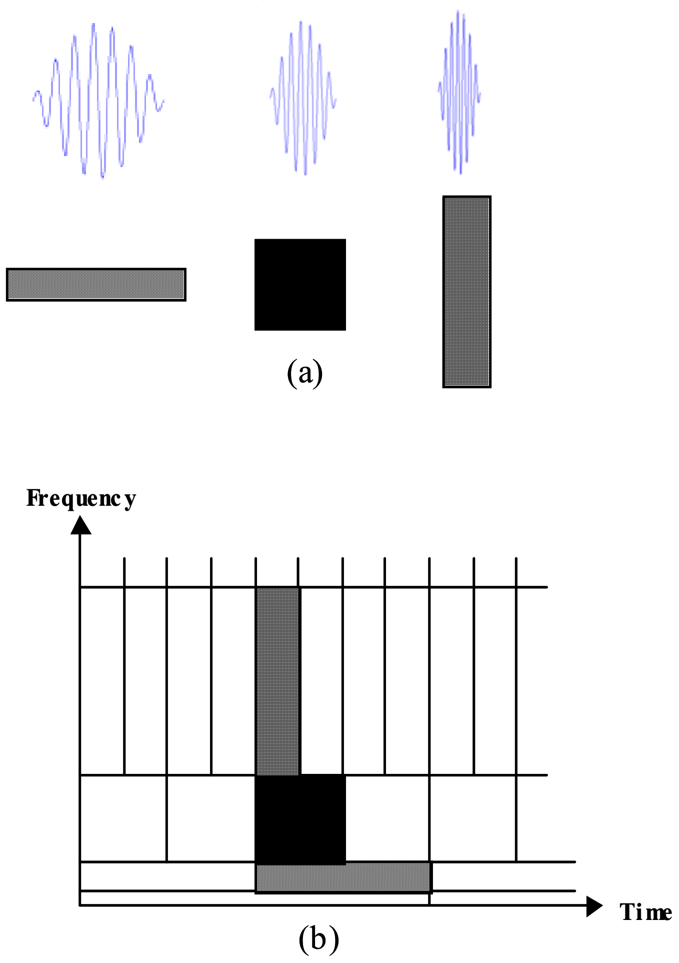


Figure 24 Basis Functions and Time–Frequency Resolution of the Wavelet Transform. (a) Basis Functions. (b) Coverage of Time–Frequency Plane (From [6].).

Here the area of each tile represents (approximately) the energy within the function (rectangular regions of the frequency plane). A characteristic of the Wavelet transform is that the tiles become shorter in time and occupy a larger frequency band as the frequency increases. By using the Wavelet techniques to develop an appropriate basis set and combining it with a quadrature mirror filter bank as illustrated in Figure 25, it is possible to decompose the waveform in such a way that the tiles have the same dimensions regardless of the frequency. By properly comparing these matrices, extracting signal features is possible using both fine frequency and fine time resolutions. Parameters, such as bandwidth, center frequency, energy distribution, phase modulation, signal duration and location in the time–frequency plane can be determined using these techniques making them valuable for intercepting receivers. [6]

Figure 25 shows the basic two–channel QMF bank. Here, the input signal $x[n]$ is first passed through a two–band analysis filter bank containing the filters, $H_0(z)$ and $H_1(z)$, which typically have lowpass and highpass frequency responses, respectively, determined by a cutoff frequency $\pi/2$ [6].

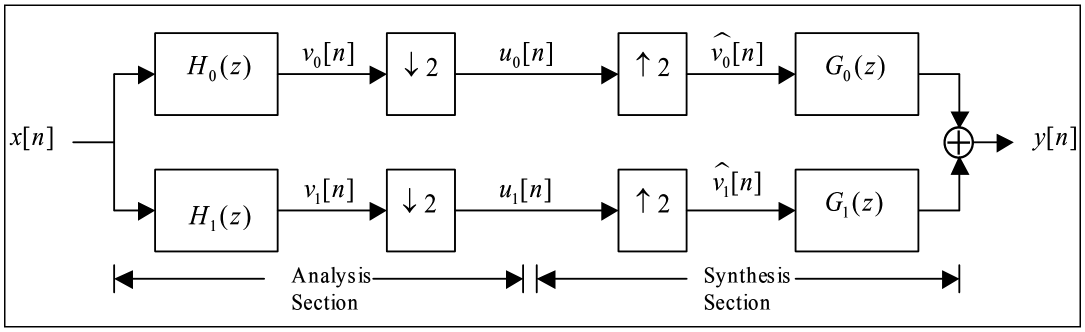


Figure 25 The Two–Channel Quadrature Mirror Filter Bank (QMFB) (From [6].).

The sub–band signals $\{v_k[n]\}$ are then down-sampled by a factor of 2 in the “signal analysis section” to be transmitted to the “signal synthesis section.” Here the signals will be up–sampled by a factor of 2 and passed through a two–band synthesis filter bank composed of the filters $G_0(z)$ and $G_1(z)$, whose outputs are then added yielding $y[n]$. The analysis and the synthesis filters in the QMF bank are chosen to ensure that the reconstructed output $y[n]$ is a reasonable replica of the input $x[n]$. Figure 26 illustrates this. The filters are further designed to provide good frequency selectivity ensuring that the sum of the power of the sub–band signals is close to the input signal power [6].

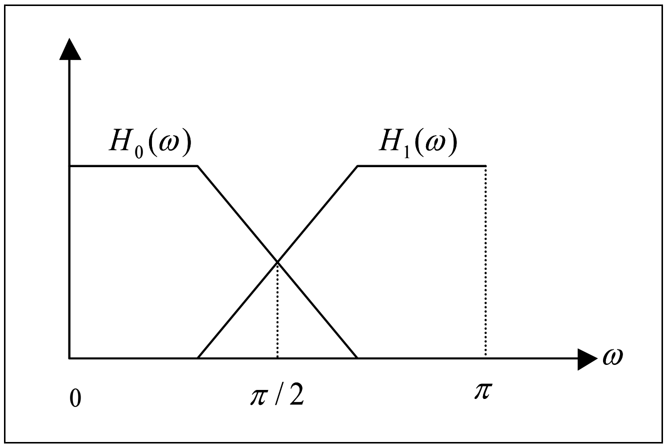


Figure 26 Typical Frequency Response of the Analysis Filters (From [6].).

The architecture of the two–channel QMFB can be extended to a tree as shown in Figure 27. This is the implementation of the QMFB being used in this thesis work.

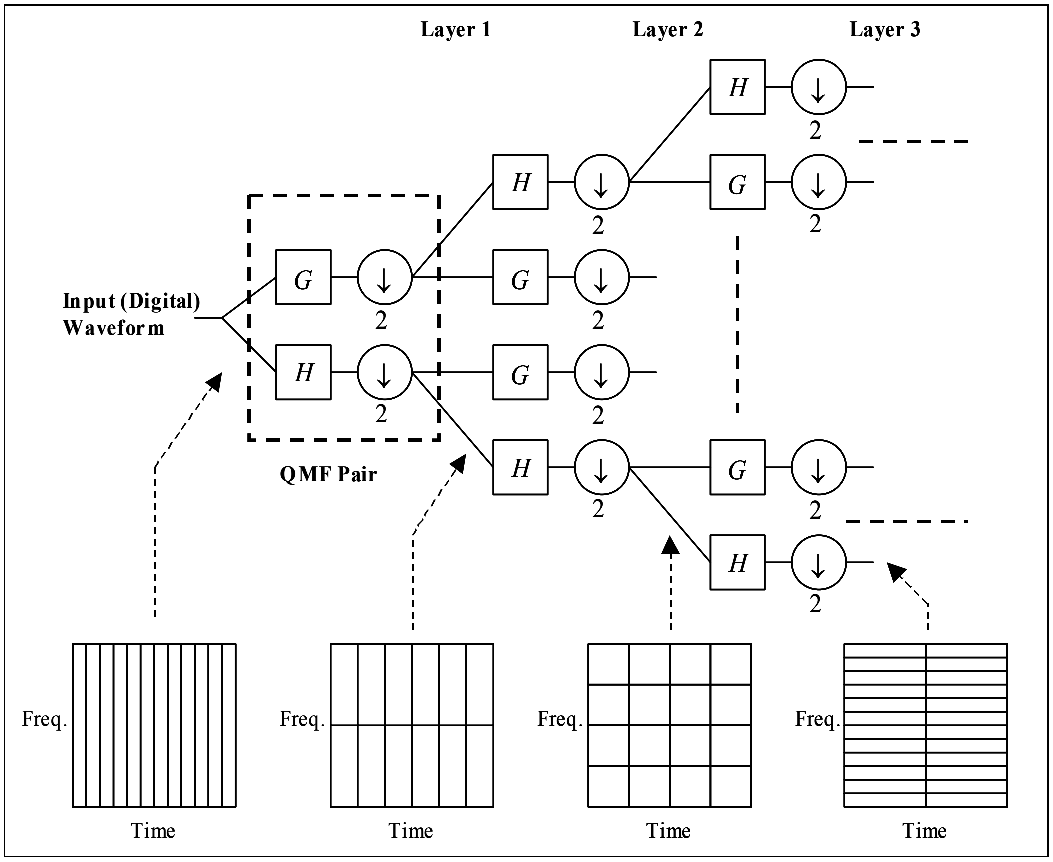


Figure 27 Quadrature Mirror Filter Bank (QMFB) Tree (From [6].).

The filters used in the QMFB must fulfill the following requirement, that is,

- They have an orthogonal decomposition so that the energy in sequences output from each QMF pair will equal the energy input.
- The output from the H filter consists of low frequency components of the input, while the output from the G filter consists of high frequency components.

One practical consequence of these requirements, as it turns out, is that when a suitable H filter is found, the G filter is obtained by negating and time reversing every other coefficient value. The filters should collect energy in approximate tiles. They must pass as much energy from inside a tile as possible, while rejecting as much as possible from outside a tile with a reasonably flat pass region.

Some filters, such as the Haar filter, meet the wavelet requirements that perfectly tile the input energy in time but, unfortunately, does not tile well in frequency. The opposite of the Haar filter, in this respect, would be the sinc filter. The correct filter is the “modified sinc filter,” which will return a good tile in time and frequency. [6]

C. WIGNER VILLE DISTRIBUTION

The implementation of the Wigner distribution used in this thesis work was made by [4, 6]. The Wigner distribution of input signal $x(t)$ is defined as

$$W(t,\omega)=\int x\left(t+\frac{\tau}{2}\right)x^{*}\left(t-\frac{\tau}{2}\right)e^{-j\omega\tau}\,d\tau \tag{1.13}$$

where t is the time variable and ω is the frequency variable. The Wigner distribution is a two–dimension function describing the frequency content of a signal as a function of time. [5]

This continuous time and frequency representation can be modified for the discrete sequence $x(l)$, where l is a discrete time index, $l=\ldots,-1,0,1,\ldots$. The discrete Wigner distribution (WD) is defined as

$$W(l,\omega)=2\sum_{-\infty}^{\infty}x(l+n)x^{*}(l-n)e^{-j2\omega n}\,. \tag{1.14}$$

If the functioned is windowed with a rectangular window function with magnitude one and some additional modification, the WD becomes [5]

$$W(l,\omega)=2\sum_{n=-N}^{N}f_{l}(n)e^{-j2\omega n}\,, \tag{1.15}$$

where

$$f_{l}(n)=x(l+n)x^{*}(l-n) \tag{1.16}$$

and where the continuous frequency variable ω is sampled by

$$\omega=\frac{\pi k}{2N},\quad k=0,1,2,\ldots,2N-1\,. \tag{1.17}$$

From equation (1.15) and (1.17) the WD becomes

$$W\left(l,\frac{\pi k}{2N}\right)=2\sum_{n=-N}^{N} f_l\left(n\right)e^{-\frac{j2\pi k}{2N}} . \tag{1.18}$$

Adjusting the limits of *n* in order to use the standard FFT algorithms, Equation (1.18) becomes

$$W\left(l,\frac{\pi k}{2N}\right)=2\sum_{n=0}^{2N-1} f_l^{'}\left(n\right)e^{-\frac{j2\pi k}{2N}} . \tag{1.19}$$

In (1.19) the kernel function has been adjusted to $f_l'\left(n\right)$, where

$$f_l'\left(n\right)=\begin{cases} f_l\left(n\right), & 0\le n\le N-1 \\ 0, & n=N \\ f_l\left(n-2N\right), & N+1\le n\le 2N-1. \end{cases} \tag{1.20}$$

The resulting WD is, therefore,

$$W\left(l,k\right)=2\sum_{n=0}^{2N-1} f_l'\left(n\right)e^{-j\frac{\pi kn}{N}} . \tag{1.21}$$

Equation (1.21) is the final WD equation used to calculate the WD of detected signals.

D. SUMMARY

Figure 28 through Figure 32 show the outputs from the three different detection methods having different input signals. Figure 28 shows a polytime code T1(2) signal, Figure 29 shows a FMCW signal with 250-Hz bandwidth, Figure 30 shows a Frank code signal, Figure 31 shows a polyphase code P4 signal and finally, Figure 32 shows the output for a BPSK signal. In each figure (a) shows the WD image, (b) shows the QMFB image and (c) shows the CYCL image. Considering the different detection methods that represent each signal and the difference between each signal using the same detection method, this thesis examines how to use the images as a means to classify the signals. That is, both time–frequency (Wigner–Ville distribution, quadrature mirror filtering) and bifrequency (cyclostationary) detection methods will be autonomously used to identify the signal modulation present.

In Chapter III the classification method including pre–processing/feature extraction will be described. Additionally the importance of the processing performed on the output images before it is suitable for classification will be stressed as well as the importance of the implemented classifiers layout and training. The chapter ends with results from the classification process using the signals described in Chapter II.

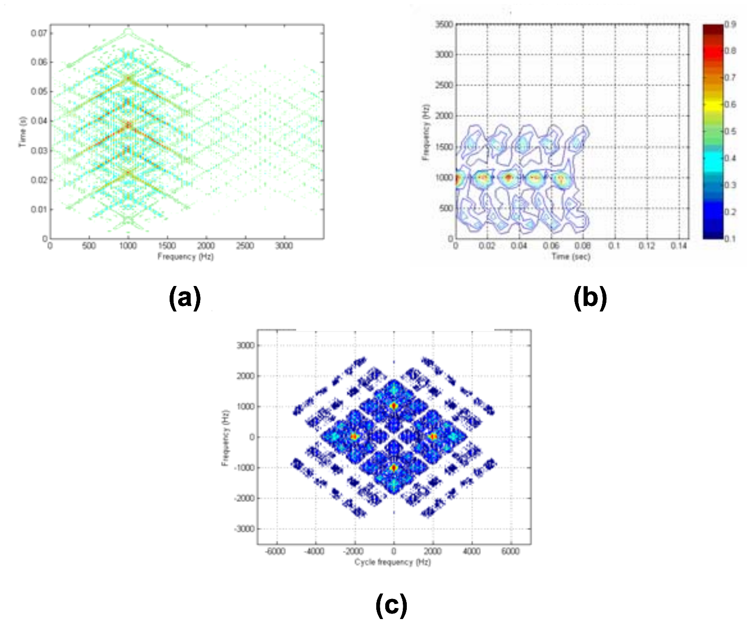


Figure 28 Illustration of the T1(2) Signal when Processed Using the Three Different Detection Methods.

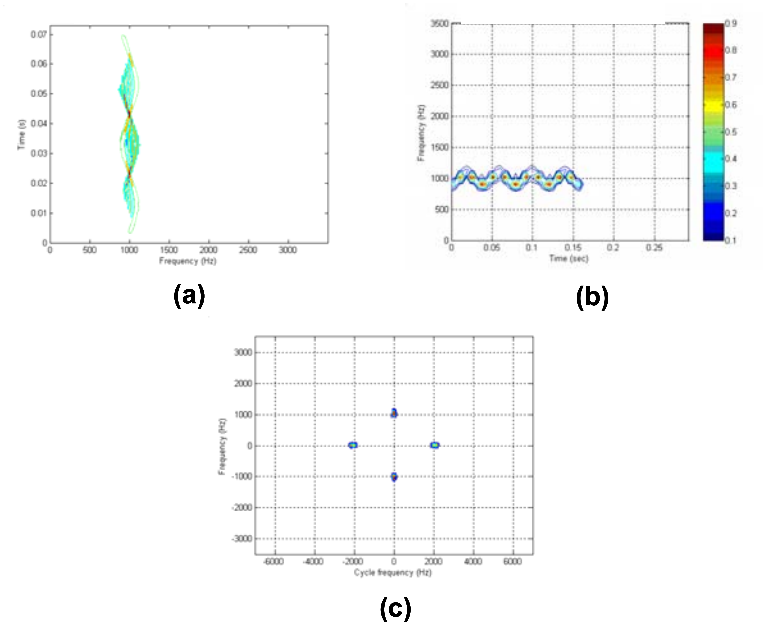


Figure 29 Illustration of the FMCW Signal when Processed Using the Three Different Detection Methods.

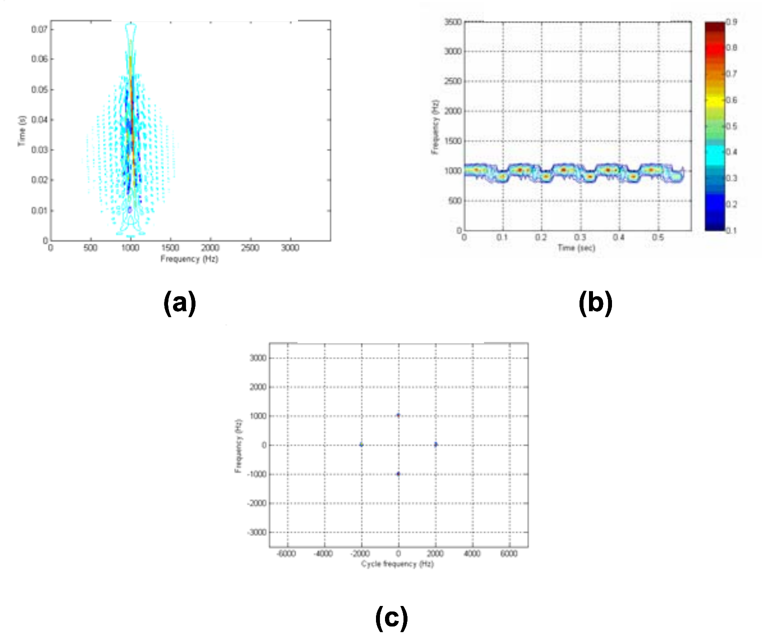
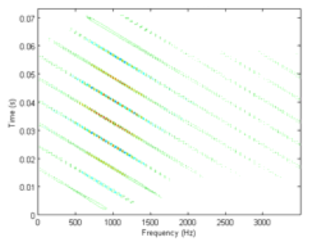
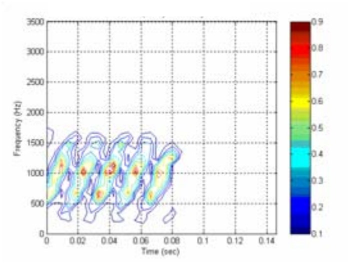
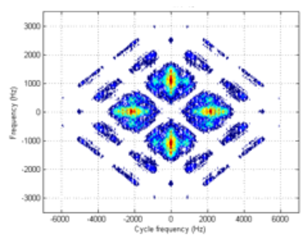


Figure 30 Illustration of the FRANK Signal when Processed Using the Three Different Detection Methods.



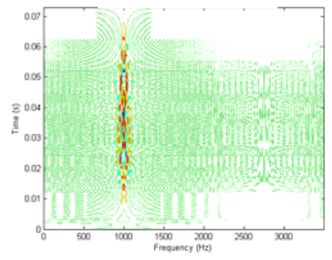


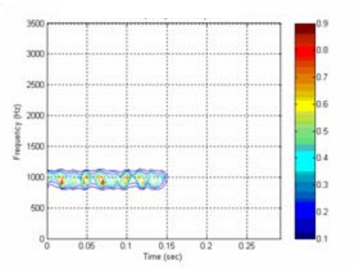
(a) **(b)**



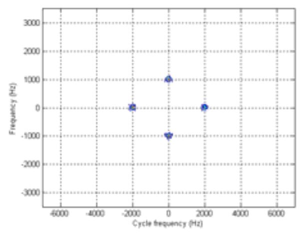
(c)

Figure 31 Illustration of the Poly Phase Signal when Processed Using the Three Different Detection Methods.





(a) **(b)**



(c)

Figure 32 Illustration of the BPSK Signal when Processed Using the Three Different Detection Methods.

IV. CLASSIFICATION

In order for a classifier to perform well, the so–called curse of dimensionality must be addressed [9]. A large number of dimensions representing the image demand a large computational capability. The demand grows exponentially with the number of dimensions [9]; therefore, the number of dimensions must be kept low. Due to this requirement the image being classified must be treated, or pre–processed, to minimize the number of dimensions the data can have. It is also important to select either linear or non–linear combinations of the original data, called features, which give a good and distinct representation of the image with a minimum of dimensions [10].

This chapter begins with a brief description of the classification process chosen for this work. Next the pre–processing method is described and illustrated with different image examples.

A. THE CLASSIFICATION METHOD

1. Use of Multiple Transform Domains

The classification method uses the output images from the three detection methods previously described. The signal inputs are limited to $f_c = 1$ kHz and $f_s = 7$ kHz. Since the signals have fixed values for the carrier frequency and the sampling frequency, the processing of the signals, in some cases, is performed using constants. This must be considered if the processing range is increased.

Recent research where features are extracted from the signal using several different transform domains have served as an inspiration for the classification method in this work. The technique is denoted Multicriteria Multitransform (MCMT). The basic idea in this approach is that the signal has a different representation within different transform domains, improving the accuracy of the classification. [11]

In earlier work three different structures have been proposed for the combined classifier; parallel–, pipeline– and hierarchical structure. In this approach, the basis for the combined classification is the different features, not the different types of classification processes. The classifiers were adapted to the selected features and the result was based upon a combined decision. An example implementation is shown in Figure 33 where features extracted from the discrete cosine transform (DCT), HAAR transform (HAAR) and singular value decomposition (SV) are used to train three separate neural networks. The "boxes" labeled Criterion 1, 2 and 3 denote the selection process (pre-processing) of significant features of the transform.

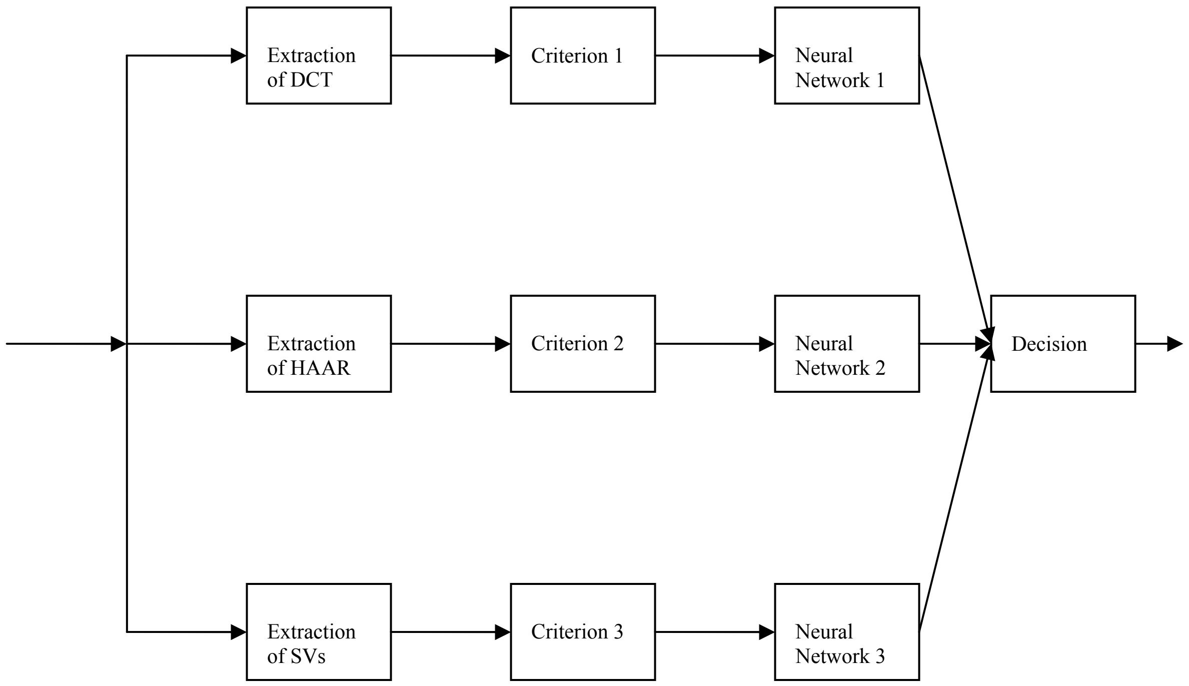


Figure 33 An Implementation Example of the MCMT Classifier (After [11].).

The proposed approach gives an opportunity to optimise the classification of data by selecting the features that are used and the weight each feature has [11].

2. Selected Classification Method

The classification method used in this work is illustrated in Figure 34. It was implemented in MATLAB and the code is included in Appendix A and B.

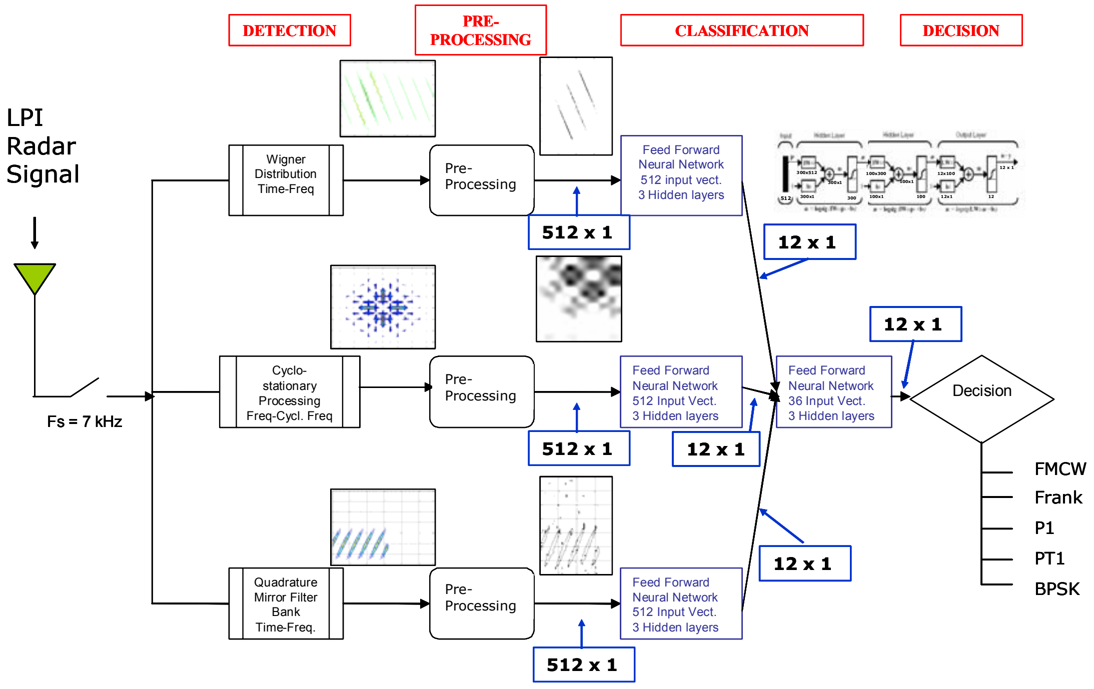

DETECTION
PRE-PROCESSING
CLASSIFICATION
DECISION
LPI Radar Signal
Fs = 7 kHz
Wigner Distribution Time-Freq
Cyclo-stationary Processing Freq-Cycl. Freq
Quadrature Mirror Filter Bank Time-Freq.
Pre-Processing
512 x 1
Feed Forward Neural Network 512 input vect. 3 Hidden layers
Feed Forward Neural Network 36 Input Vect. 3 Hidden layers
12 x 1
Decision
FMCW
Frank
P1
PT1
BPSK

Figure 34 Illustration of Classification Method.

Figure 34 shows the detection methods that are used in separate "pipes". The output images from each detection method are pre–processed in order to extract distinct information from the image and in order to correct the problem of too many inputs to the classifier [9]. The result of the pre–processing is a column vector with 512 rows. After pre–processing the column vector are fed into the selected classifier, which in this work is a feed–forward neural network. Each classifier gives a 12×1 column vector indicating which of the modulation types given in Table 7 are present in the input signal. The modulation types are numbered 1 to 12 with the number being used to describe the results for each type of signal. The reason for 12 rows is to create a network structure that is working with all 12 signal types generated in the LPIG toolbox [1].

The vector from each pipe is combined with the output vectors from the other two pipes and is fed as a 36×1 column vector into the last neural network which in turn gives a 12×1 column vector with the final suggestion of which modulation type is contained in the input signal.

Table 7 Signals Place in Input Matrices.

SIGNAL REPRESENTATION	
1	BPSK
2	Costas code
3	FMCW
4	Frank code
5	P1
6	P2
7	P3
8	P4
9	T1
10	T2
11	T3
12	T4

For different modulation types, the images produced by a single detection method are sometimes very similar. This makes it hard to get an acceptable classification result. Therefore, each signal is classified using three detection methods in parallel. Each method gives a different output for each signal type and the parallel process provided the results from all three detection methods to the last neural network. With the last neural network properly trained, the final classification decision has a higher probability of being correct than if the classification is made solely upon a single detection method.

Critical issues for the classification process to be successful are the pre–processing [9] and the construction of the neural networks [10]. A majority of the time put into this thesis work has been within these two areas and more work is needed to obtain an optimum solution.

B. FEATURE EXTRACTION/PRE–PROCESSING METHOD

The selection of the pre–processing and feature extraction methods are normally one of the most significant factors in determining the performance of the final system [10]. It is critical to minimize the dimensionality of the problem and to extract the information out of a noisy image. There are two types of pre–processing techniques, transform technique or morphological technique. It is also common to use combinations of these types [9]. In this work tests with different types of pre–processing were made and the final selection of pre–processing/feature extraction methods are shown in Table 8. The methods are shown in the table in the order that they are applied to the images. Each of the methods will be more closely described later in this chapter.

Table 8 Used Pre–processing/Feature Extraction Methods.

PRE-PROCESSING METHODS			
	Wigner Ville Distribution	**Cyclostationary Processing**	**Quadrature Mirror Filter Bank**
1	Cropping (256*256)	Cropping (128*128)	Cropping (255*255)
2	Input Normalization	Input Normalization	Input Normalization
3	Adaptive filtering	Adaptive filtering	Adaptive filtering
4	Feature Filtering	Feature Filtering	Feature Filtering
5	Dilation	Dilation	Dilation
	Closing	Closing	Closing
6	Distance transform	Image resizing	Image resizing
7	Image resizing	Reshaping to 512*1	Reshaping to 512*1
8	Reshaping to 512*1		

The sources for this work are mainly References [8, 9, 12] but also recent research of References [10, 11] has provided input to this process. The final selection of methods was based upon different trials with the complete classification process.

The pre–processing/feature extraction methods selected were different depending on the detection method used. The reason is the substantial difference in the output images versus the three detection methods. In obtaining the best result from the classifier it is important that the feature represent each signal, and that the feature is distinct between the signal types.

1. Pre-Processing

Throughout the pre–processing section, an image representing each of the five modulation types is used to illustrate the effect of the different techniques. The images representing the signals are shown in Table 9. The data for each signal is given in Table 3.

Table 9 Signals Illustrating the Pre–processing Techniques.

SIGNAL IMAGES USED TO ILLUSTRATE PRE-PROCESSING	
BPSK	B_1_7_7_1_s
FMCW	F_1_7_500_20_s
Frank Code	FR_1_7_4_1_s
P4	P4_1_7_16_1_s
T1	PT1_1_7_2_4_s

a. Image Cropping

Image cropping might not belong to pre–processing by definition, but it is treated as such here due to its role in this work. With the help of the cropping function, the image was reduced solely to the area of interest. No definition of cropping is given here since it is only an extraction of a subset from the original image.

The implementation of cropping in this work was through the MATLAB function “imcrop.” An example of how the function is applied to the different signals is shown in Figure 35 through Figure 39. Each figure shows the image representations for WD, QMFB and CYCL being cropped. The figures differ by type of signal as:

- Figure 35 shows the B_1_7_7_1_s signal,
- Figure 36 shows the F_1_7_500_20_s signal,
- Figure 37 shows the FR_1_7_4_1_s signal,
- Figure 38 shows the P4_1_7_16_1_s signal,
- Figure 39 shows the PT1_1_7_2_4_s signal.

The different areas of interest noticeably depend on signal type. The images generated by the CYCL method are cropped to a smaller size due to the repeated pattern in this method.

There are methods developed that seek the optimum features for classification [12] but those methods were not implemented in this work since the detection methods give a predetermined output thereby making it possible to decide the area of interest in advance.

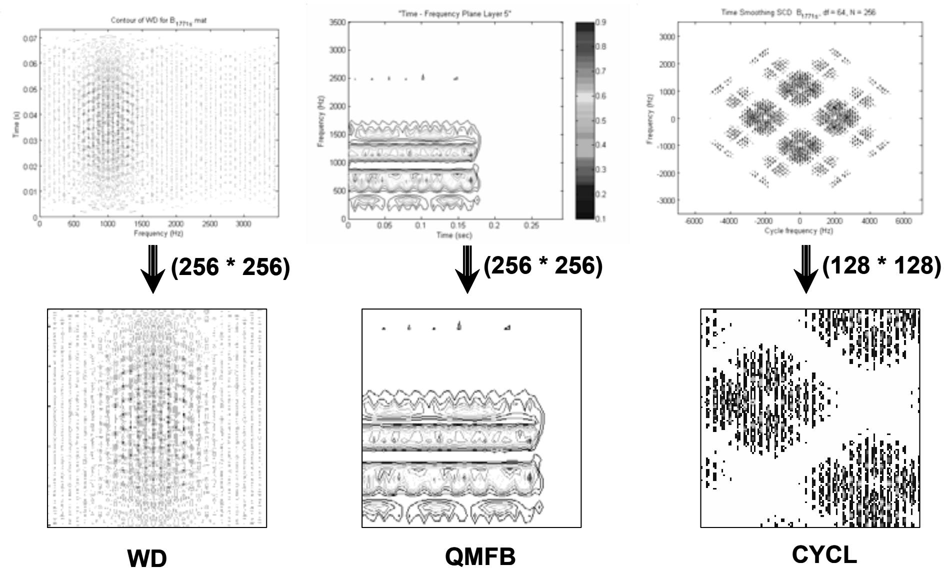


Figure 35 Cropping of the Output Images for the B_1_7_7_1_s Signal.

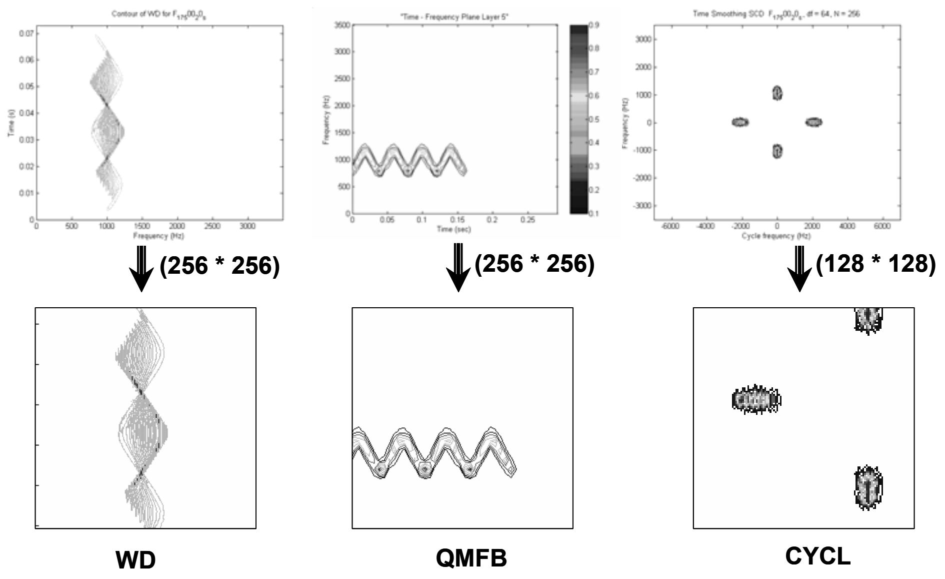


Figure 36 Cropping of the Output Images for the F_1_7_500_20_s Signal.

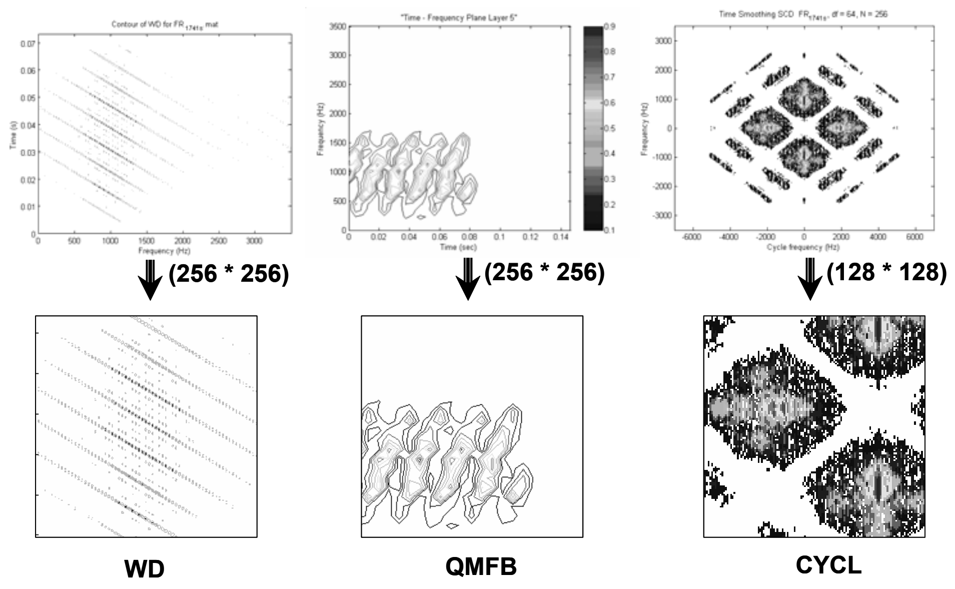


Figure 37 Cropping of the Output Images for the FR_1_7_4_1_s Signal.

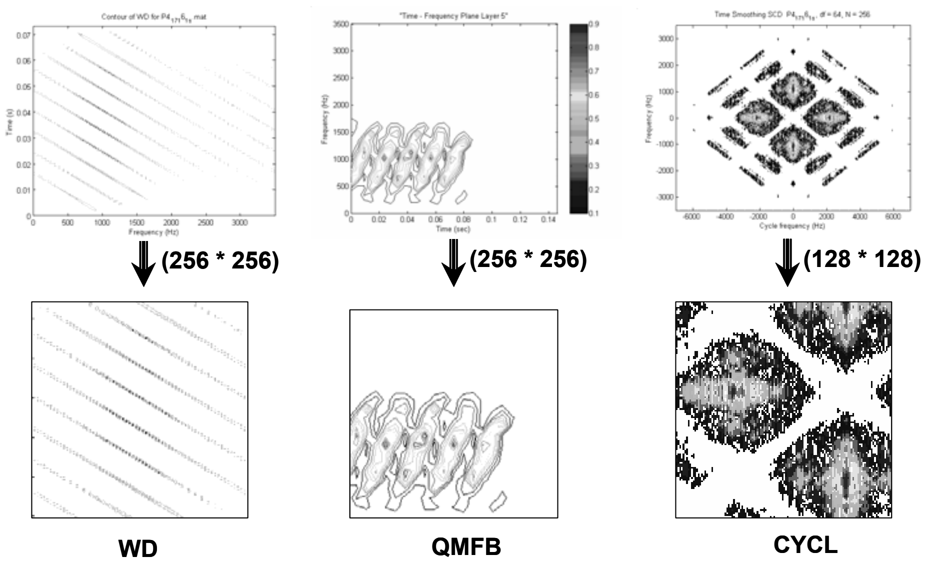


Figure 38 Cropping of the Output Images for the P4_1_7_16_1_s Signal.

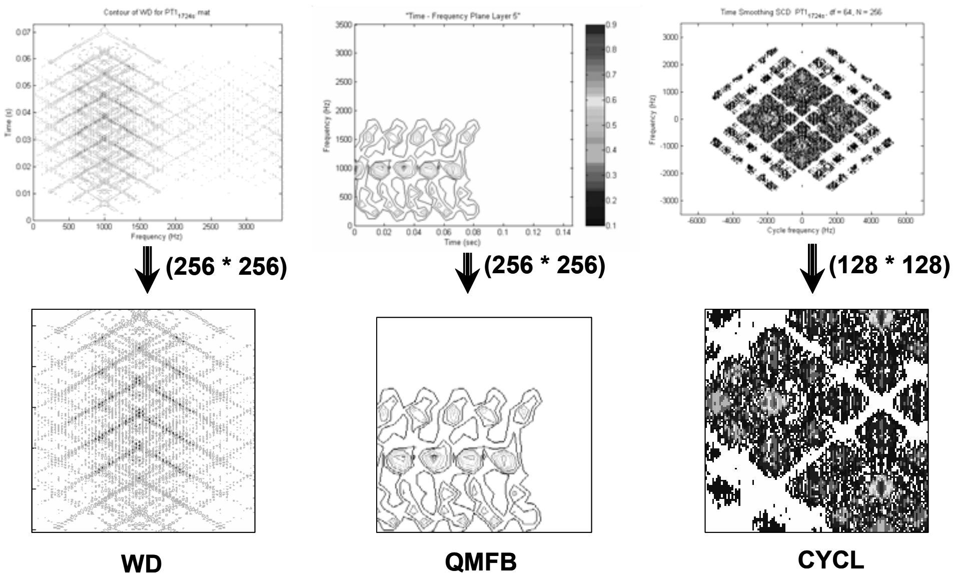


Figure 39 Cropping of the Output Images for the PT1_1_7_2_4_s Signal.

b. Input Normalization

In a situation where the dynamic range of the image is different, the data having a larger value might have a larger influence in the classification process. One solution to avoid this is to normalize the input data using the estimates of the mean, $\overline{x}_k$, and the variance, σ_k^2. If given *N* available data sets of the *k*-th feature, $\hat{x}_{ik}$, then

$$\overline{x}_k = \frac{1}{N}\sum_{i=1}^{N} x_{ik}, \quad k = 1,2,\ldots,l, \tag{4.1}$$

$$\sigma_k^2 = \frac{1}{N-1}\sum_{i=1}^{N}\left(x_{ik} - \overline{x}_k\right)^2, \tag{4.2}$$

and

$$\hat{x}_{ik} = \frac{x_{ik} - \overline{x}_k}{\sigma_k}. \tag{4.3}$$

The above method is linear and the result is a normalized image with zero mean and a uniform variance. Another method would be to scale the data linearly within certain ranges like $\left[0,1\right]$ or $\left[-1,1\right]$ described by [9]. A third alternative is to use a non–linear method and scale the data within a certain interval. An example of this is the so–called *softmax scaling,* which is a two–step process [9]

$$y = \frac{x_{ik} - \overline{x}_k}{r\sigma_k}, \tag{4.4}$$

and

$$\hat{x}_{ik} = \frac{1}{1+\exp\left(-y\right)}. \tag{4.5}$$

This function limits the data in the range [0, 1]. For small values of y the function is approximately linear with respect to x_{ik}. The range for this is user–defined by the factor *r*. Values away from the mean are squashed exponentially.

In this work, normalizing the input image is accomplished through the MATLAB function "imadjust." This function remaps the intensity of the images within a given range, here [0 1]. This is an application of the softmax method described above. The effect of normalizing can be seen in Figure 40 through Figure 44 where the effect of normalization applied on the WD, QMFB and CYCL representation is shown for the signals of interest. The difference between the figures is which signal representation being shown where:

- Figure 40 shows the effect on a B_1_7_7_1_s signal,
- Figure 41 shows the effect on a F_1_7_500_20_s signal,
- Figure 42 shows the effect on a FR_1_7_4_1_s signal,
- Figure 43 shows the effect on a P4_1_7_16_1_s signal,
- Figure 44 on a PT1_1_7_2_4_s signal.

Normalization has a noticeable effect on the image produced by the Wigner Distribution.

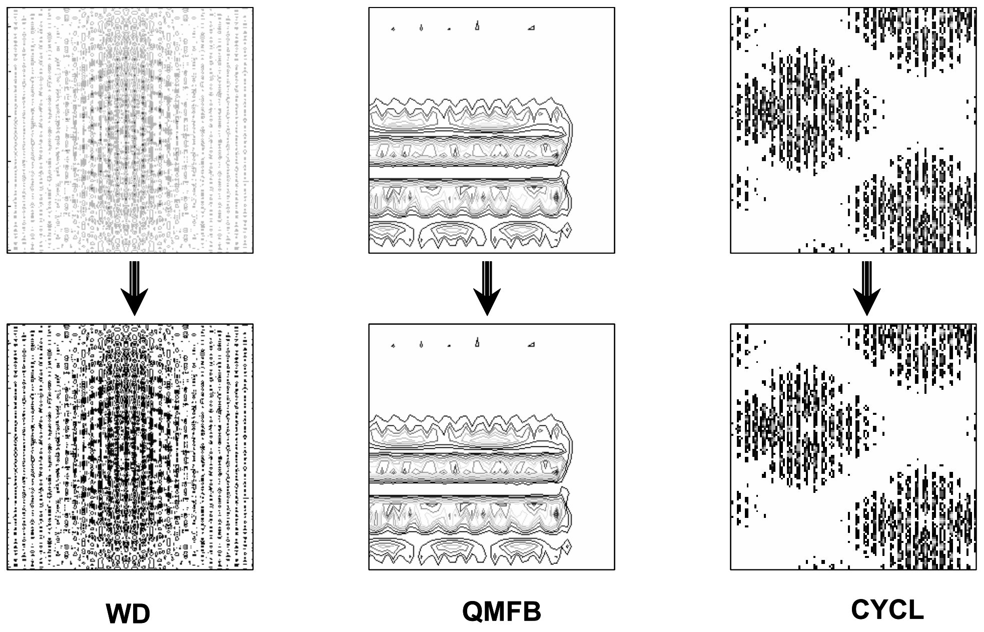


Figure 40 Normalization of the B_1_7_7_1_s Signal.

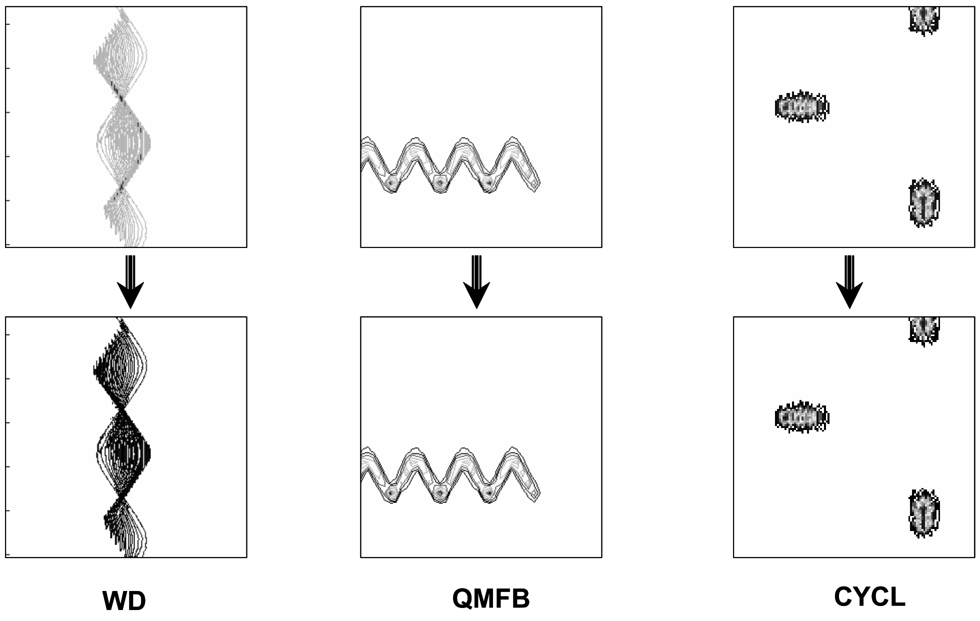


Figure 41 Normalization of the F_1_7_500_20_s Signal.

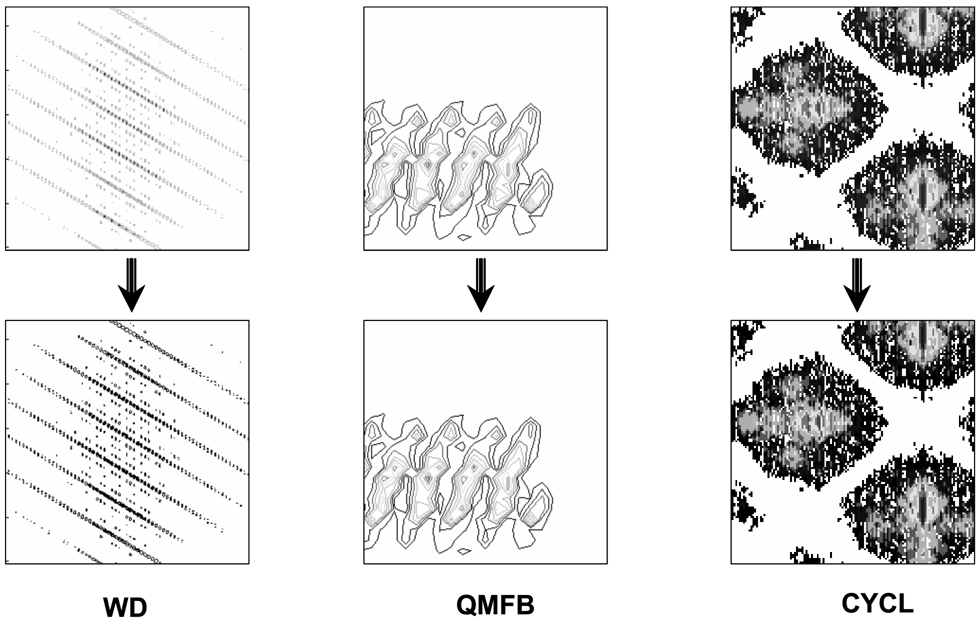


Figure 42 Normalization of the FR_1_7_4_1_s Signal.

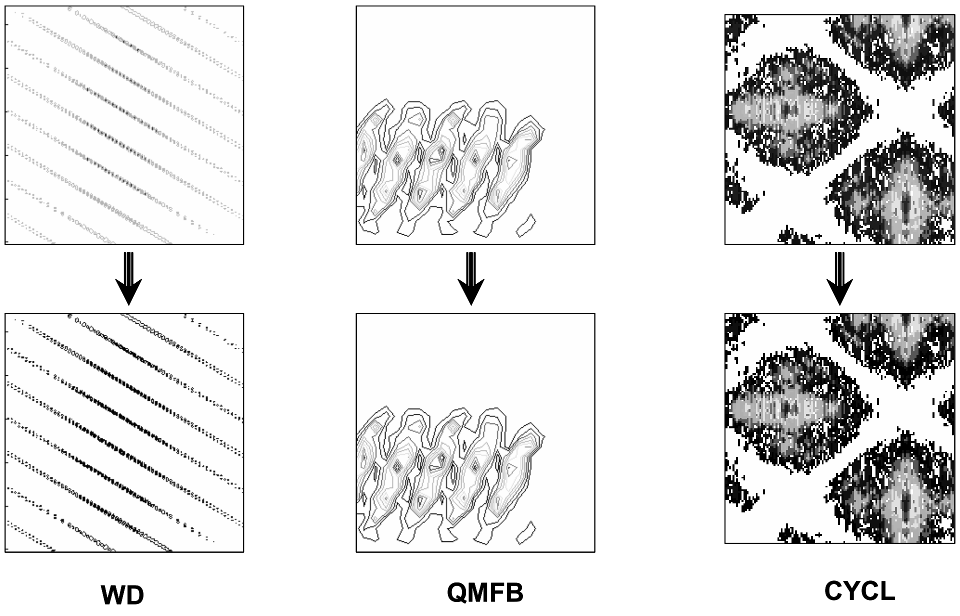


Figure 43 Normalization of the P4_1_7_16_1_s Signal.

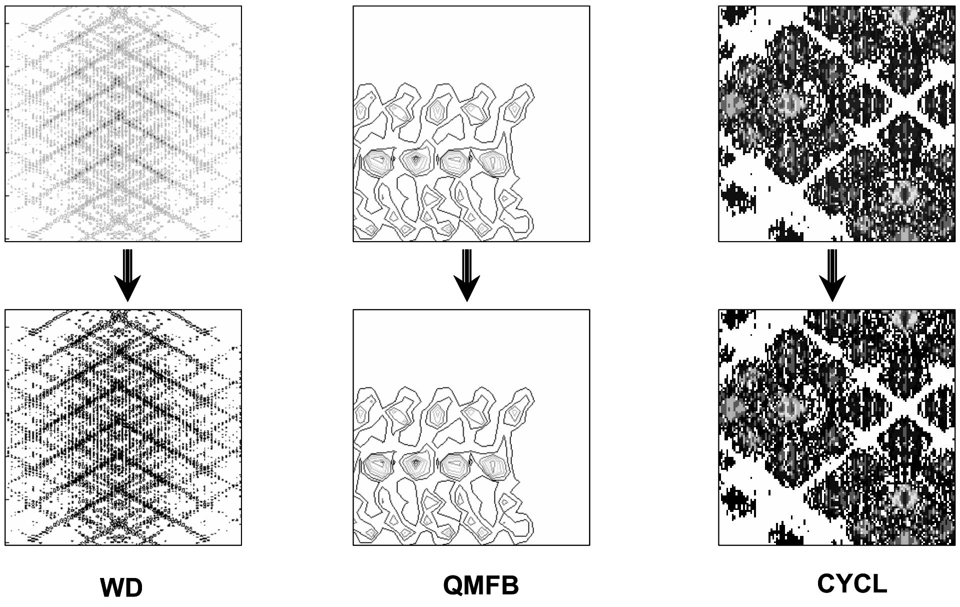


Figure 44 Normalization of the PT1_1_7_2_4_s Signal.

2. Feature Extraction

a. Adaptive Filtering

An adaptive filter is a filter that changes behavior based on the statistical characteristics of the image inside the region of the filter. Compared with other filter types, an adaptive filter performs well, but with the negative aspect of being complex [13].

If a filter is to operate on region S_{xy}, the response on a point (x, y) is dependent on the actual value $g(x, y)$, the variance of the noise σ_η^2, the local mean m_L of the pixels in S_{xy} and finally the local variance σ_L^2 of the pixels. The filter operates by providing little smoothing if the differences between the two variances are large and more smoothing if the variance difference is small. An expression for the estimated value of the image, $\hat{f}(x, y)$, is

$$\hat{f}(x,y) = g(x,y) - \frac{\sigma_\eta^2}{\sigma_L^2}\left[g(x,y) - m_L\right]. \tag{4.6}$$

The quantity σ_η^2 must be estimated but the rest is measured in the image.

In this work the implementation of an adaptive filter is illustrated in Figure 45 through Figure 49. Each figure illustrates the effect of the adaptive filtering on the images being generated by the WD, the QMFB and the CYCL. As earlier, the difference between the figures is which type of signal they represent where:

- Figure 45 shows the effect on the B_1_7_7_7_s signal,
- Figure 46 shows the effect on the F_1_7_500_20_s signal,
- Figure 47 shows the effect on the FR_1_7_4_1_s signal,
- Figure 48 shows the effect on the P4_1_7_16_1_s signal,
- Figure 49 shows the effect on the PT1_1_7_2_4_s signal.

The information in the WD images is reduced and the QMFB images are smoothed providing a more homogeneous image. To perform this filtering the MATLAB function used was “wiener2” which provides a linear filter with behavior as described above.

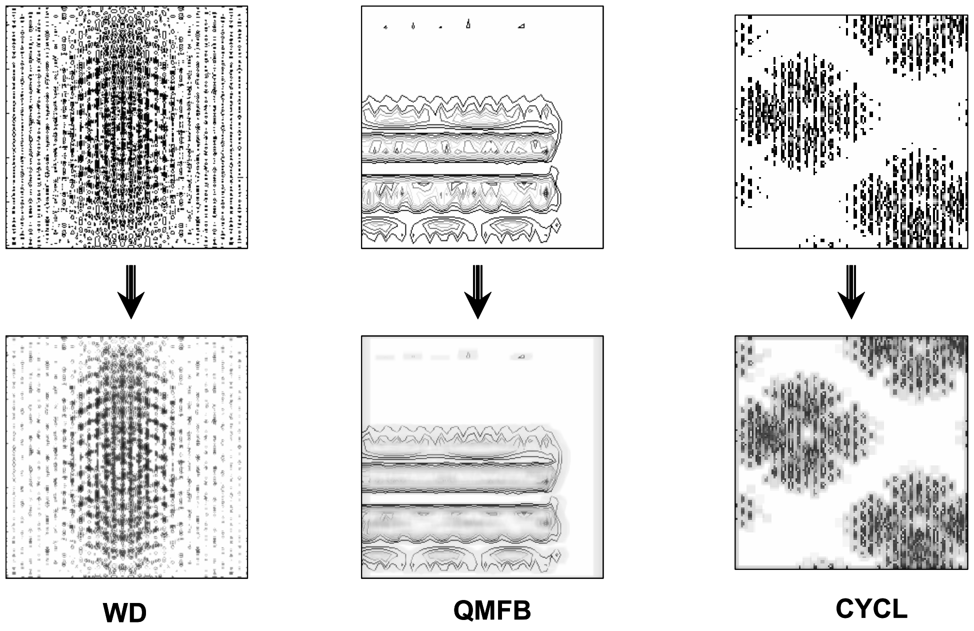


Figure 45 Adaptive Filtering of the B_1_7_7_7_s Signal.

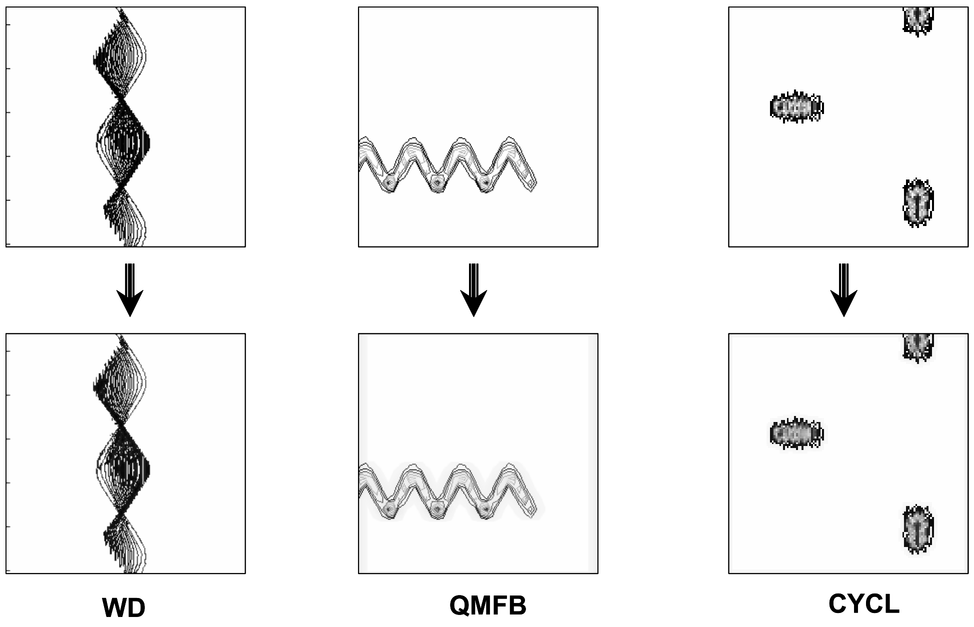


Figure 46 Adaptive Filtering of the F_1_7_500_20_s Signal.

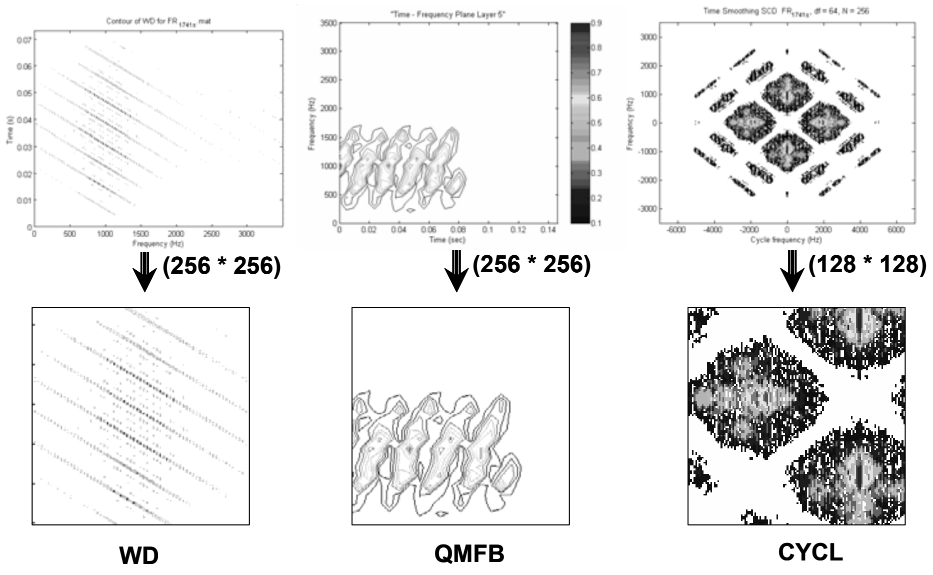


Figure 47 Adaptive Filtering of the FR_1_7_4_1_s Signal.

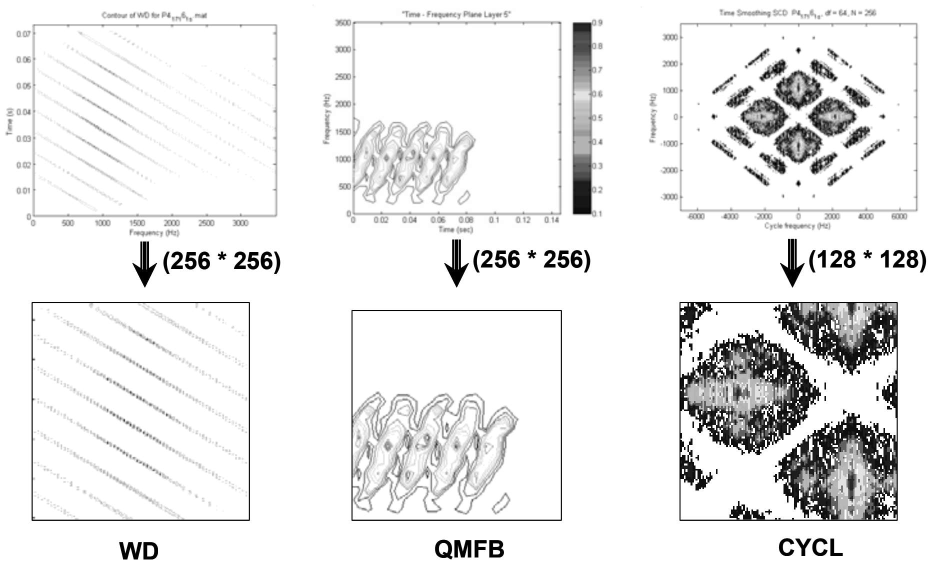


Figure 48 Adaptive Filtering of the P4_1_7_16_1_s Signal.

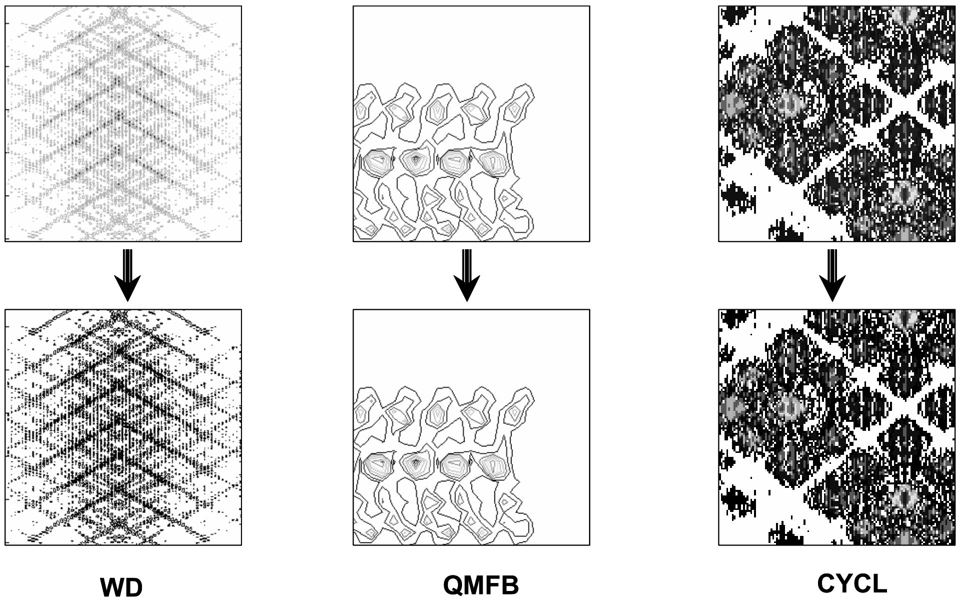


Figure 49 Adaptive Filtering of the PT1_1_7_2_4_s Signal.

b. Feature Filtering

After the adaptive filter, the “imtophat” and “imbothat” functions were used to extract “peak” information from the image. These methods return the peaks or valleys of objects that fit a redefined structuring element. The result from the application of these functions was then used in the rest of the feature extraction processing. Results of the operation are seen in Figure 50 through Figure 54, where each figure shows the feature filtering on the WD, the QMFB and the CYCL image representation of the selected signals. The figures differ as:

- Figure 50 shows the B_1_7_7_7_s signal,
- Figure 51 shows the F_1_7_500_20_s signal,
- Figure 52 shows the FR_1_7_4_1_s signal,
- Figure 53 shows the P4_1_7_16_1_s signal,
- Figure 54 shows the PT1_1_7_2_4_s signal.

The structuring element, which is optimized for slant lines, used in the operation is illustrated in Figure 55. The reduction of information in the images is evident but a distinct representation still exists for each signal. If no limitations in number of inputs to a classifier existed, the images generated below would have been suitable for the classification process.

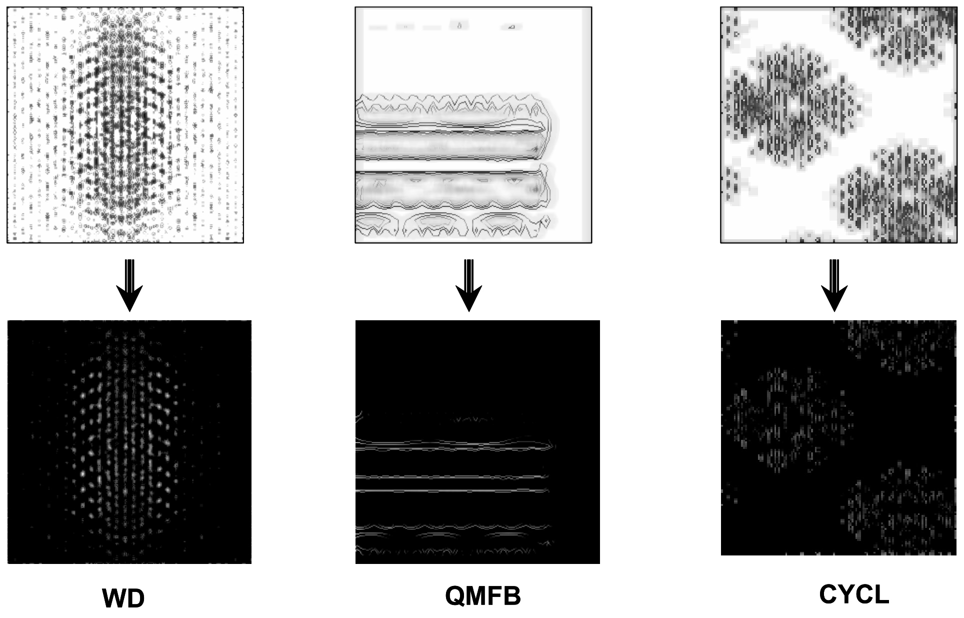


Figure 50 Information Extraction from the B_1_7_7_7_s Signal.

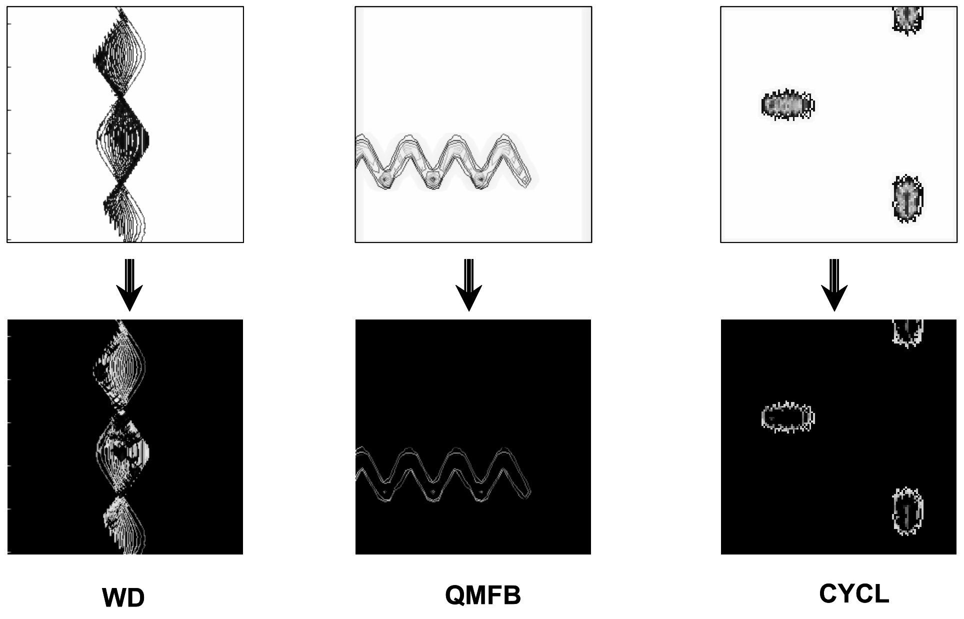


Figure 51 Information Extraction from the F_1_7_500_20_s Signal.

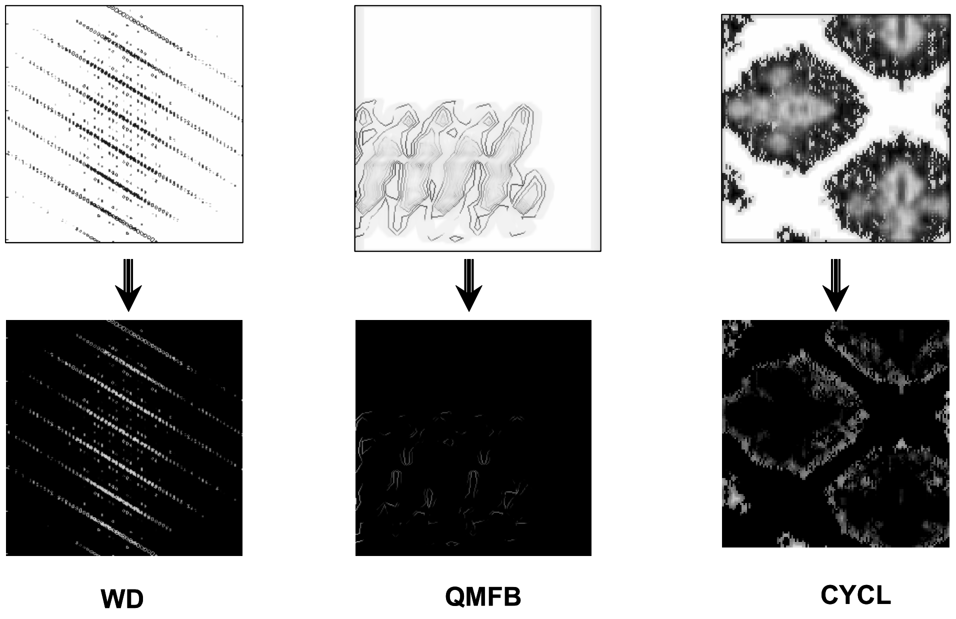


Figure 52 Information Extraction from the FR_1_7_4_1_s Signal.

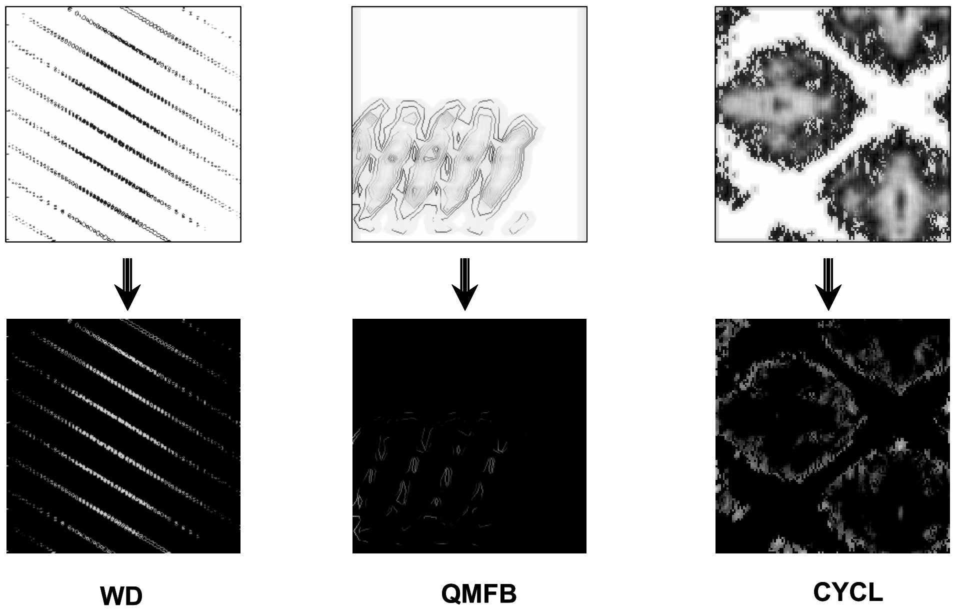


Figure 53 Information Extraction from the P4_1_7_16_1_s Signal.

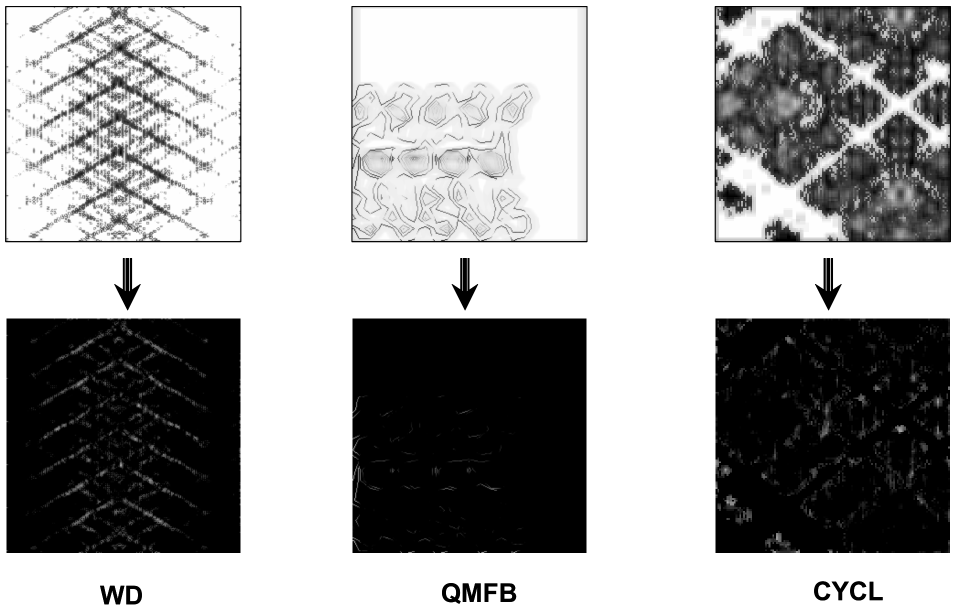


Figure 54 Information Extraction from the PT1_1_7_2_4_s Signal.

0	0	0	0	1	0	0	0	0
0	0	0	1	1	1	0	0	0
0	0	1	1	1	1	1	0	0
0	1	1	0	0	0	1	1	0
1	1	0	0	0	0	0	1	1
1	0	0	0	0	0	0	0	1

Figure 55 Structuring Element used in “imtophat” and “imbothat”.

c. *Dilation and Erosion*

Dilation and Erosion are two functions that are fundamental to morphological operations. They work as a base for many more complex operations while being essential to improve image quality [13].

DILATION, which is denoted $A \oplus B$, is defined as

$$A \oplus B = \left\{ z \middle| \left(\hat{B}\right)_z \cap A \neq \varnothing \right\}. \tag{1.7}$$

Here A and B are sets in Z^2. The equation is based on obtaining the reflection of B about its origin and then shifting this reflection by z. The dilation is the set of displacements, z, where A and B overlap at least with one element. The set B is known as the structuring element and can be viewed as a convolution mask [13]. This is because one can view the dilation as “flipping” B about its origin and then sliding it over the image A.

Figure 56 below shows a simple set, A, together with the structuring element, B and the structuring elements reflection (the dilation). Here the dashed line describes the original image while the solid line describes the resulting image.

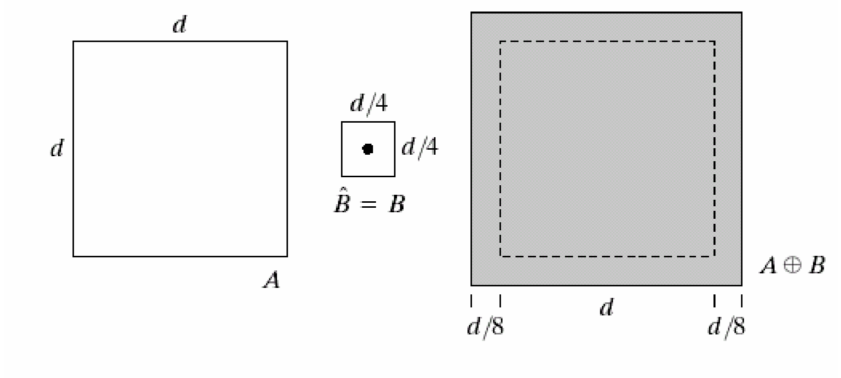


Figure 56 Dilation of A by B (From [13].).

Figure 57 below shows the result of using a different element B on the image A. In this figure the connection between the structuring element and the resulting image can be easily seen.

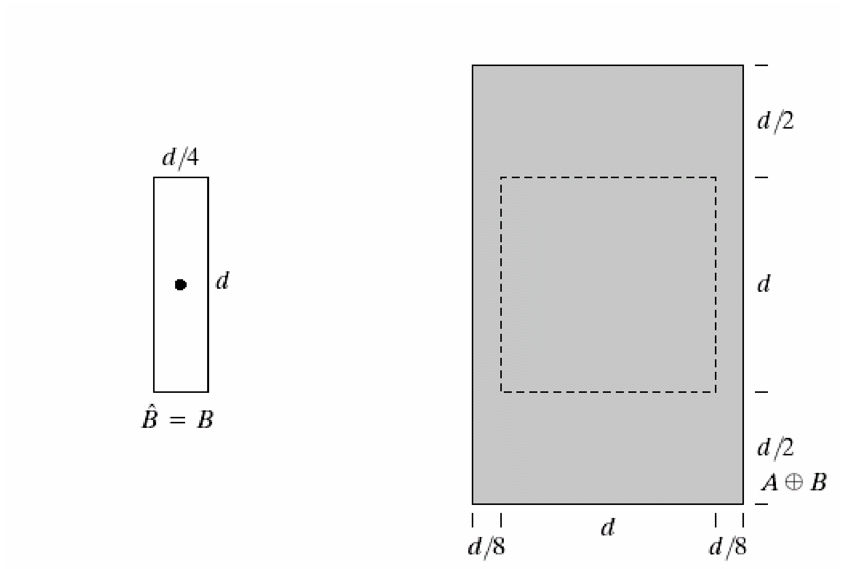
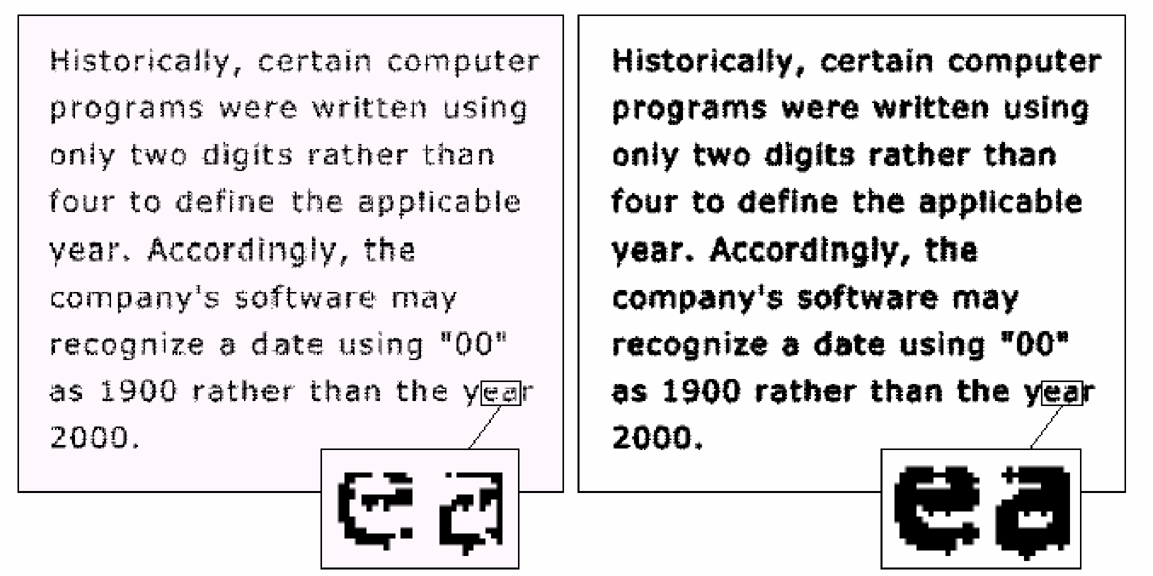


Figure 57 Dilation of A by B (From [13].).

Dilation in its simplest application is used to close gaps in images. Figure 58 exemplifies the poor text being treated with a cross–shaped structuring element shown at the bottom of the figure, resulting in the broken segments being joined together.



Historically, certain computer programs were written using only two digits rather than four to define the applicable year. Accordingly, the company's software may recognize a date using "00" as 1900 rather than the year 2000.

Historically, certain computer programs were written using only two digits rather than four to define the applicable year. Accordingly, the company's software may recognize a date using "00" as 1900 rather than the year 2000.

0	1	0
1	1	1
0	1	0

Structuring element

Figure 58 Dilation Treatment of Poor Text (From [13].).

EROSION, denoted $A \ominus B$ is defined as

$$A \ominus B = \left\{ z \,\middle|\, (B)_z \subseteq A \right\}. \tag{1.8}$$

This means that the erosion of A by B is the set of all points z such that B translated by z is contained in A. Figure 59 and Figure 60 below show the process using the same structuring element as in Figure 56 and Figure 57 above.

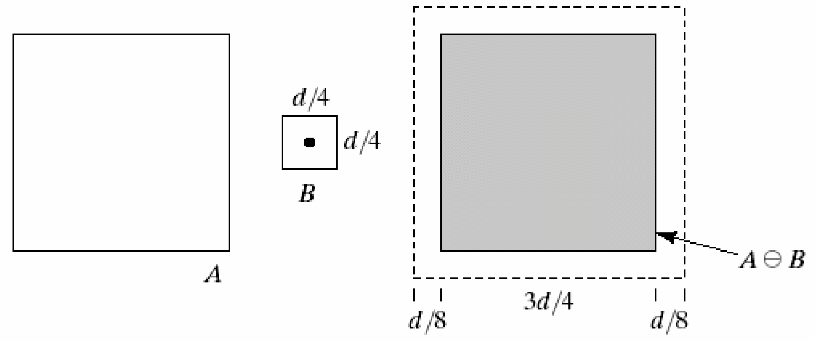


Figure 59 Erosion of A with Structuring Element B (From [13].).

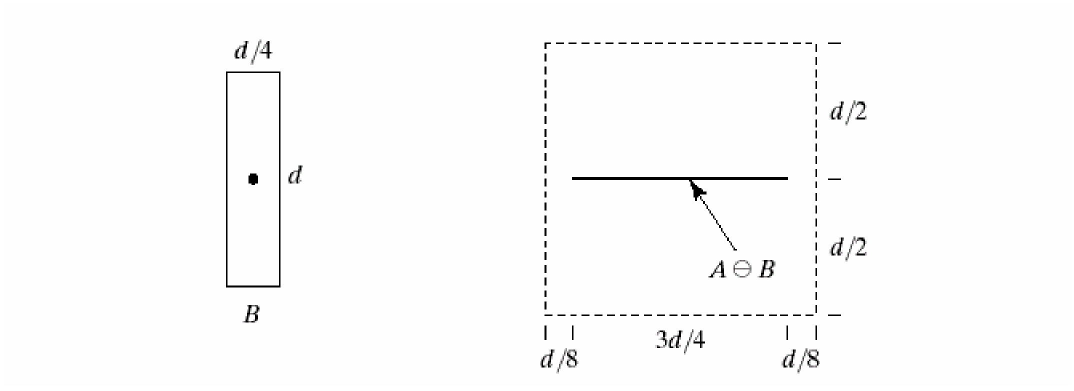


Figure 60 Erosion of A with Different B (From [13].).

Erosion can be used to eliminate irrelevant information out of a binary image [13]. An example of this is shown in Figure 61 below. The initial image contains squares of different sizes (size 1, 3, 5, 7, 9 and 15 pixels). The second image shows the result after erosion using a square structuring element of size 13 pixels. The third image shows the result after dilution of the second image reusing the same structuring element

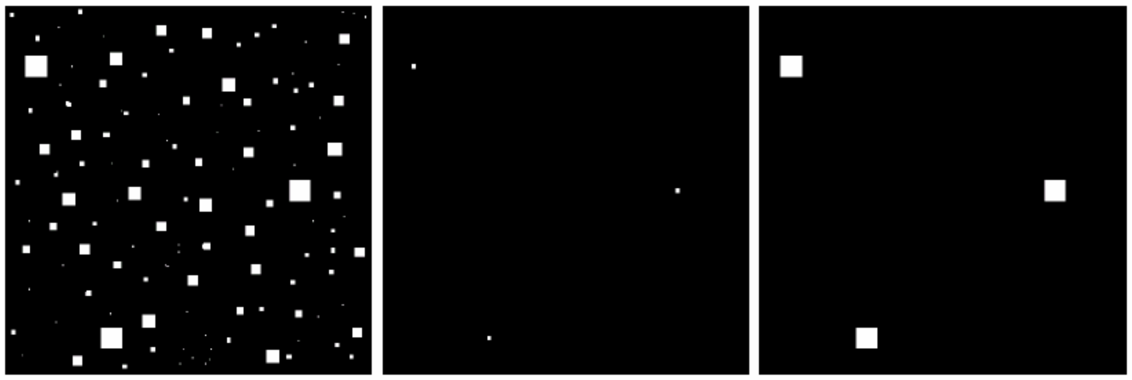


Figure 61 Erosion of Image Containing Irrelevant Information (From [13].).

In this work the dilation process was implemented after the images were normalized, and filtered with the peak information being extracted. The result of the dilation process on the images produced by WD, QMFB and CYCL are shown in Figure 62 through Figure 66. The difference between the figures is which signal they show where

- Figure 62 shows the B_1_7_7_7_s signal,
- Figure 63 shows the F_1_7_500_20_s signal,
- Figure 64 shows the FR_1_7_4_1_s signal,
- Figure 65 shows the P4_1_7_16_1_s signal,
- Figure 66 shows the PT1_1_7_2_4_s signal.

The dilation process creates a "smearing" effect on the shapes in the image that corresponds to the shape of the structuring element.

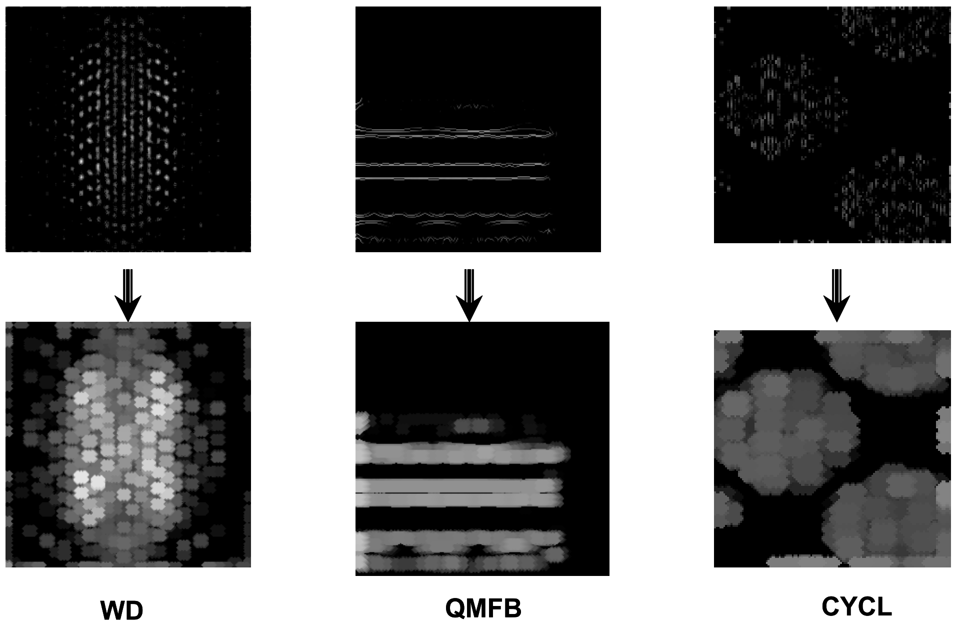


Figure 62 The Dilation Process Applied to the B_1_7_7_7_s Signal.

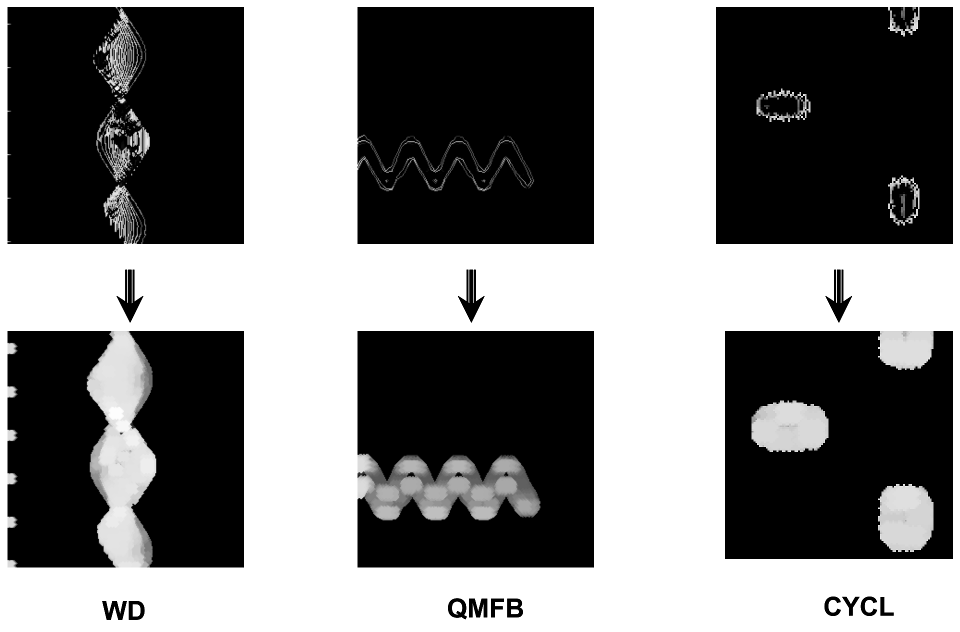


Figure 63 The Dilation Process Applied to the F_1_7_500_20_s Signal.

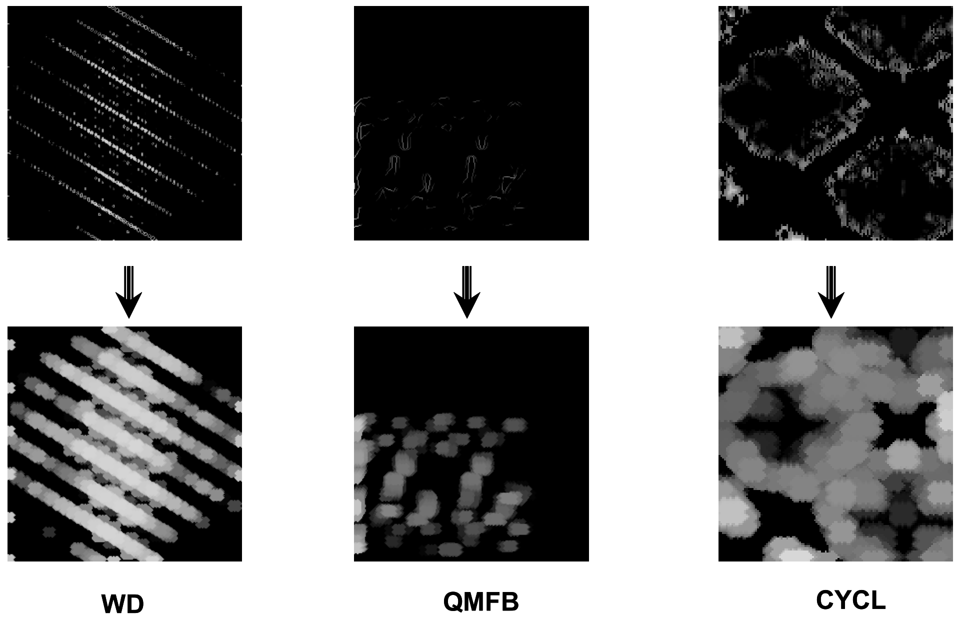


Figure 64 The Dilation Process Applied to the FR_1_7_4_1_s Signal.

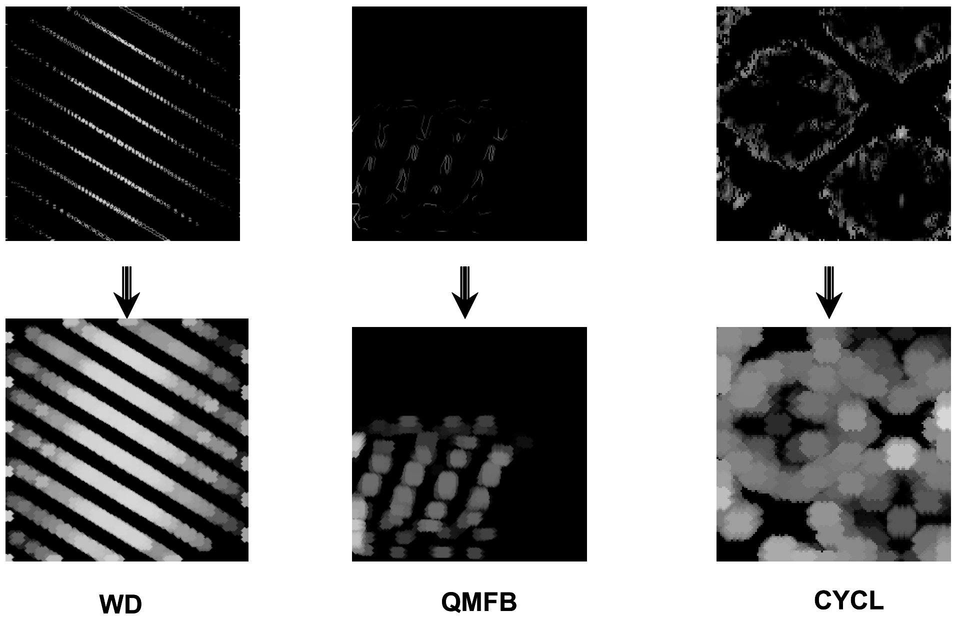


Figure 65 The Dilation Process Applied to the P4_1_7_16_1_s Signal.

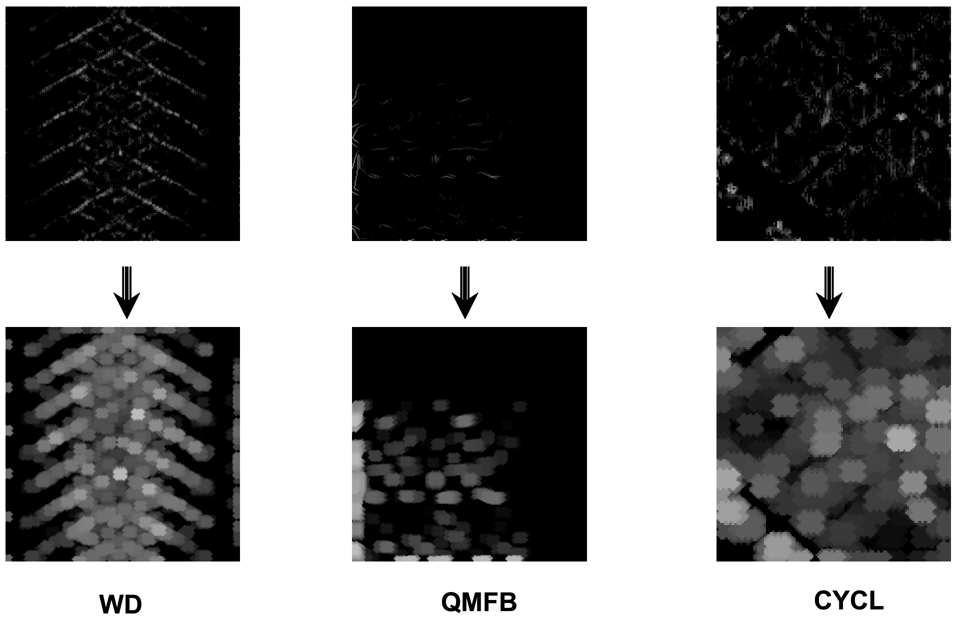


Figure 66 The Dilation Process Applied to the PT1_1_7_2_4_Signal s.

Figure 67 shows the structuring element used in this operation. This structuring element is optimized for horizontal, vertical and leaning lines.

0	1	1	0	1	0	1	1	0
0	0	0	1	1	1	0	0	0
0	0	1	1	1	1	1	0	0
0	1	1	0	1	0	1	1	0
1	1	0	0	1	0	0	1	1
1	0	0	0	1	0	0	0	1

Figure 67 Structuring Element for the Dilation Process.

d. Opening and Closing

Two other important morphological operations are opening and closing. Opening has a smoothing effect on the contour of an object, breaks narrow lines and eliminates thin protrusions. Closing also has a smoothing effect but has the opposite effect of opening in that it fuses thin gulfs and fills gaps in the contour [13].

OPENING of the set A by structuring element B, which is denoted as $A \circ B$, is defined as

$$A \circ B = \left(A \ominus B \right) \oplus B . \tag{1.9}$$

As indicated by the definition, opening is the erosion of A by B followed by a dilation by B. The opening operation can be seen as a ball, formed by the structuring element B, which is rolling inside the limits formed by the image A. The boundary of $A \circ B$ is then established by the points of B that reach the farthest into the edges of A [13]. Opening can also be expressed as a fitting process defined by

$$A \circ B = \cup \left\{ \left(B \right)_z \middle| \left(B \right)_z \subseteq A \right\} \tag{1.10}$$

which, in other words, means opening of A by B is obtained by taking the union of all translates of B that fit into A. Figure 68 below shows the process applied to a triangle using a circular structuring element.

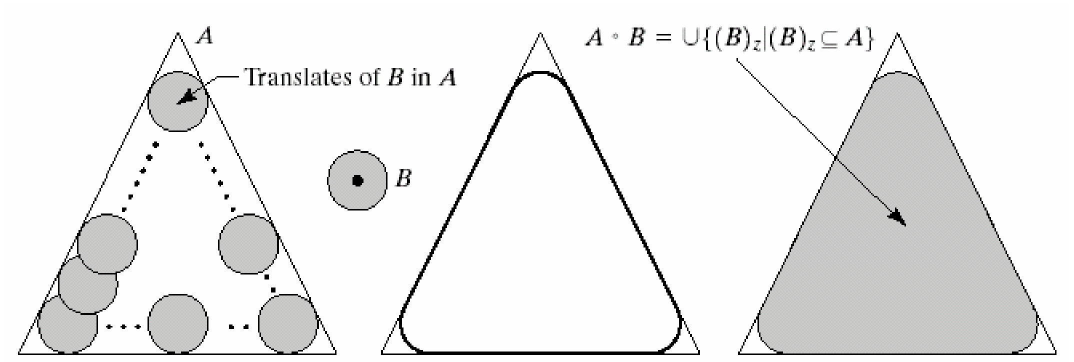


Figure 68 Opening with a Circular Structuring Element B (From [13].).

CLOSING, denoted by $A \bullet B$, is defined as

$$A \bullet B = (A \oplus B) \ominus B. \tag{1.11}$$

In other words, the closing of A by B is the dilation of A by B, followed by the erosion using B again. This can also be geometrically described with the rolling ball, but this time the ball roll on the outside as seen in Figure 69 below.

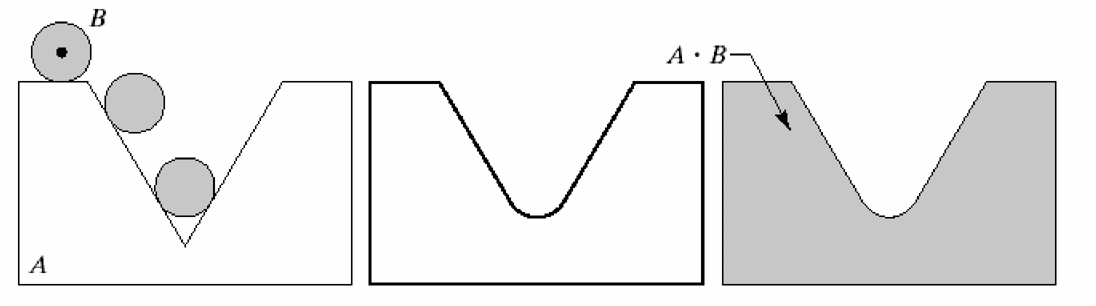


Figure 69 Closing with a Circular Structuring Element B (From [13].).

The implementation of closing is shown in Figure 70 through Figure 74 for the images produced by WD, QMFB and CYCL. The figures differ in the signal that is illustrated where:

- Figure 70 shows the B_1_7_7_7_s signal,
- Figure 71 shows the F_1_7_500_20_s signal,
- Figure 72 shows the FR_1_7_4_1_s signal,
- Figure 73 shows the P4_1_7_16_1_s signal,
- Figure 74 shows the PT1_1_7_2_4_s signal.

The closing smoothes small features in the image which might affect the classification process by erasing certain features. The effect of feature reduction was especially noticeable in the WD representation of the B_1_7_7_7_s signal.

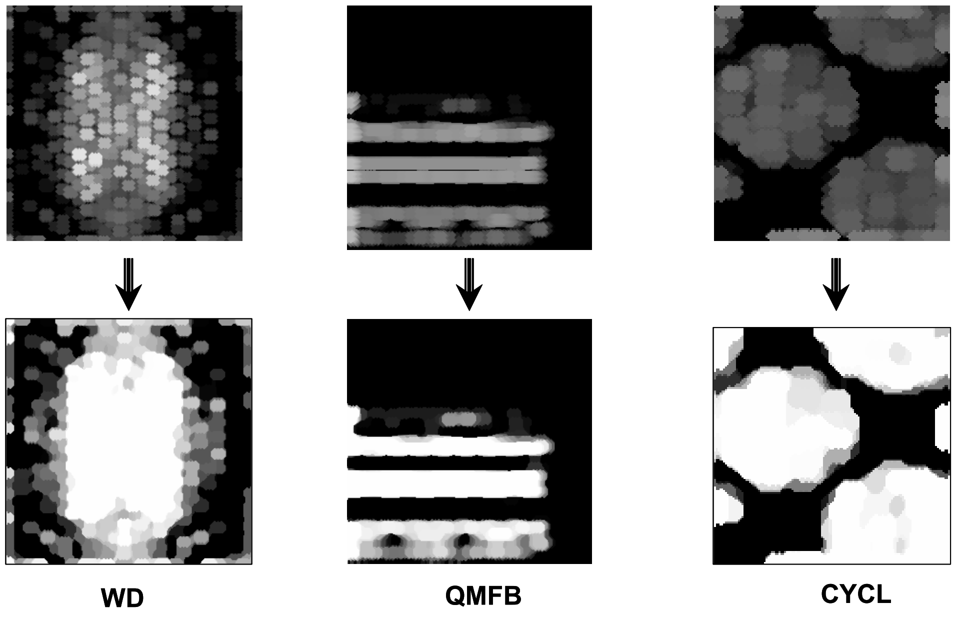


Figure 70 The Closing Process Applied on the B_1_7_7_7_s Signal.

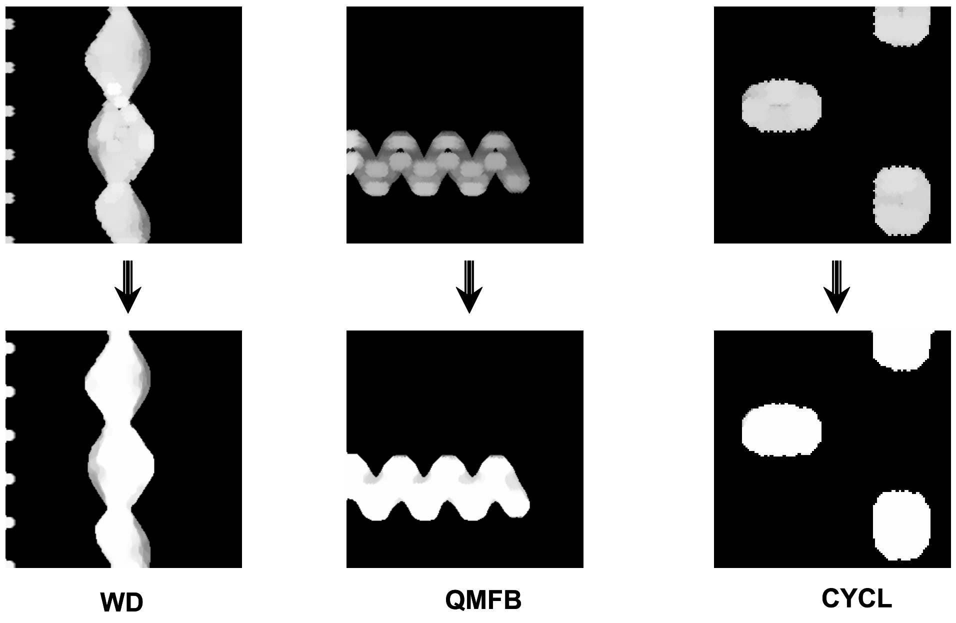


Figure 71 The Closing Process Applied on the F_1_7_500_20_s Signal.

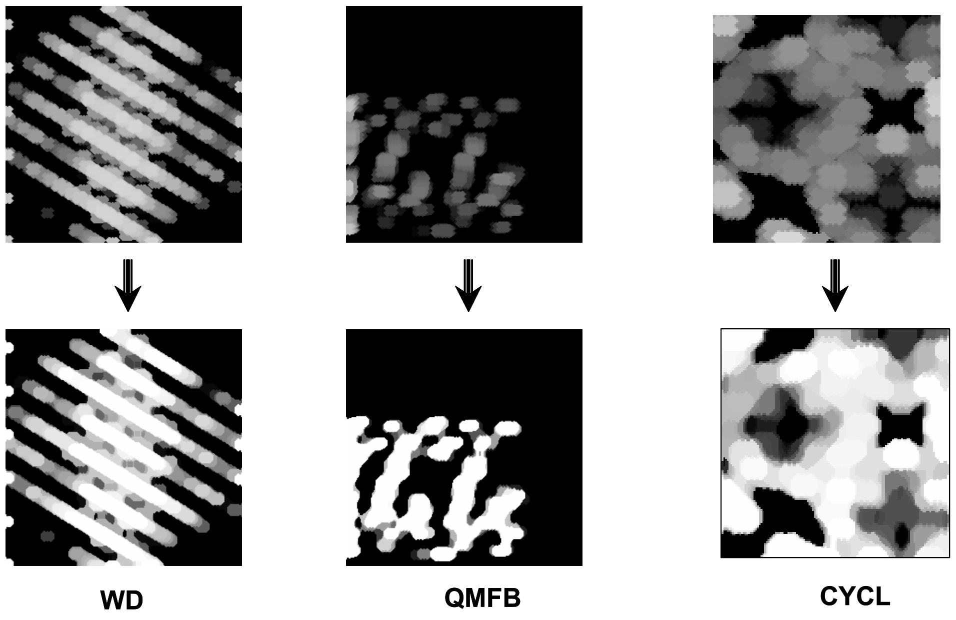


Figure 72 The Closing Process Applied on the FR_1_7_4_1_s Signal.

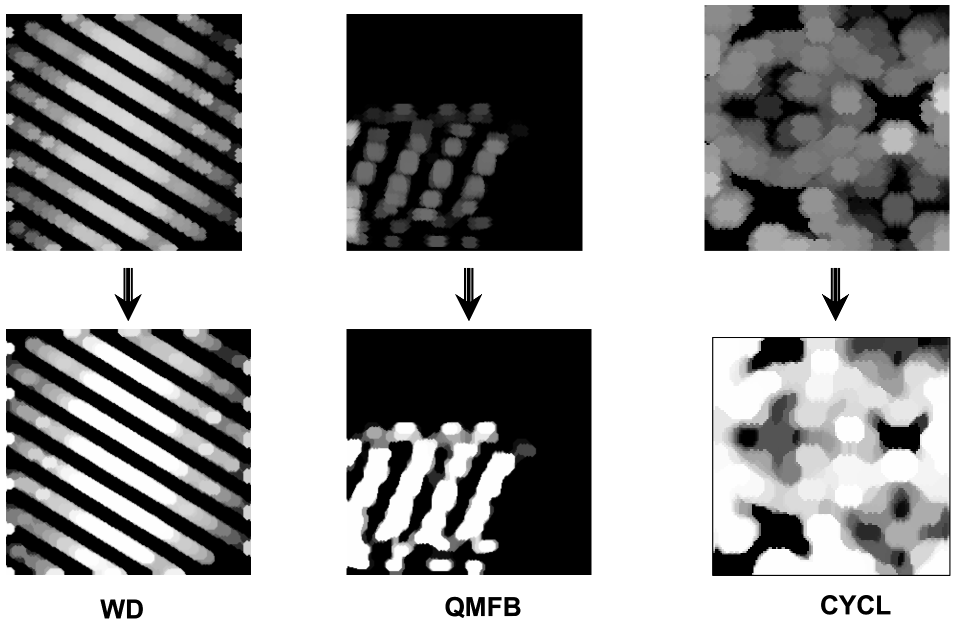


Figure 73 The Closing Process Applied on the P4_1_7_16_1_s Signal.

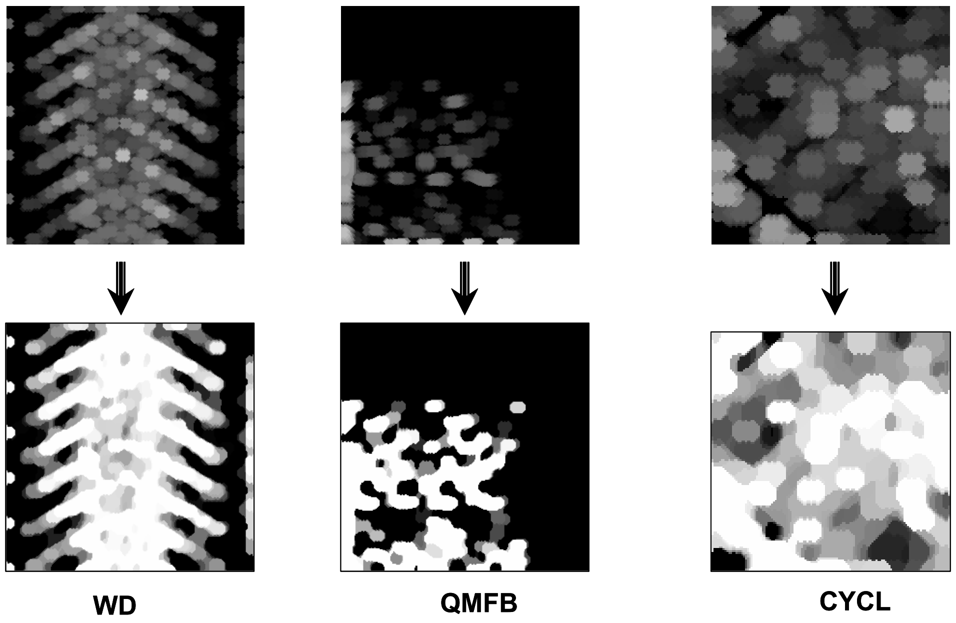


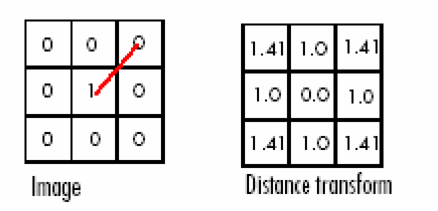
Figure 74 The Closing Process Applied on the PT1_1_7_2_4_s Signal.

e. Distance Transform

The distance transform provides a metric of measure for separating the points in an image. If the pixels p and q have coordinates (x, y) and (s, t) respectively, the Euclidean distance function, D_e, is defined as

$$D_e(p,q) = \left[(x-s)^2 + (y-t)^2\right]^{\frac{1}{2}}. \tag{4.12}$$

Figure 75 illustrates the Euclidean distance transform.



0	0	0
0	1	0
0	0	0

1.41	1.0	1.41
1.0	0.0	1.0
1.41	1.0	1.41

Figure 75 Euclidean Distance Transform. (From [14].).

The implementation of the distance transform in this work is illustrated in Figure 76 through Figure 80 where only the transform for the WD representation is shown. The distance transform is not applied to the QMFB and CYCL representations of the signals since the reason for pre–processing/feature extraction is to produce a distinct image representation useful for classification purposes. The resulting image of the distance transform is binary meaning that grayscale information used for classification purposes is erased. In the QMFB and CYCL representations of the signals, the grayscale information was deemed important; therefore, the distance transform only applied to the WD representation of the signals. The figures differ as:

- Figure 76 shows the B_1_7_7_7_s signal,
- Figure 77 shows the F_1_7_500_20_s signal,
- Figure 78 f shows the FR_1_7_4_1_s signal,
- Figure 79 shows the P4_1_7_16_1_s signal,
- Figure 80 shows the PT1_1_7_2_4_s signal.

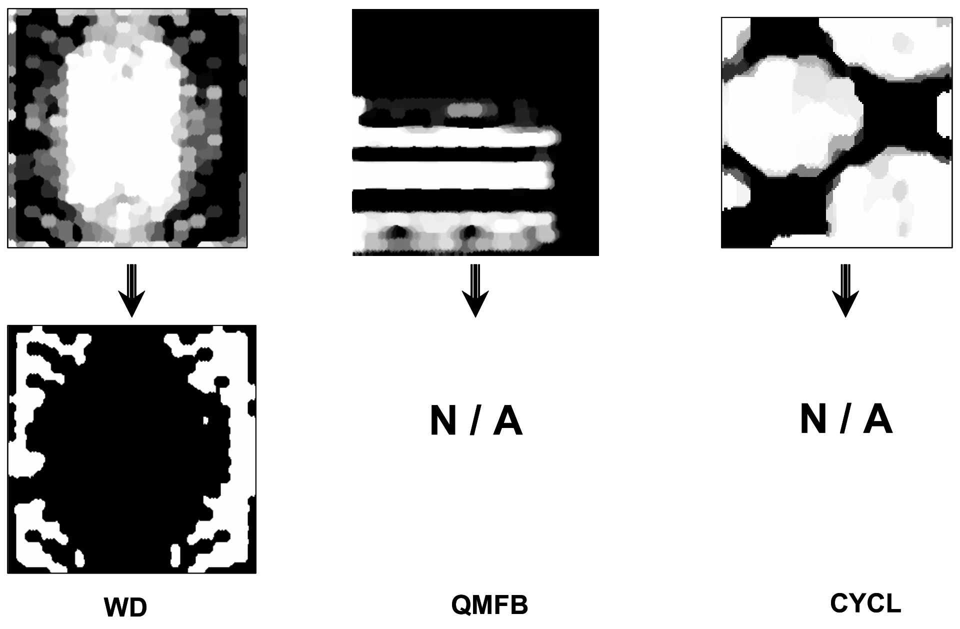


Figure 76 Distance Transform of the B_1_7_7_7_s Signal.

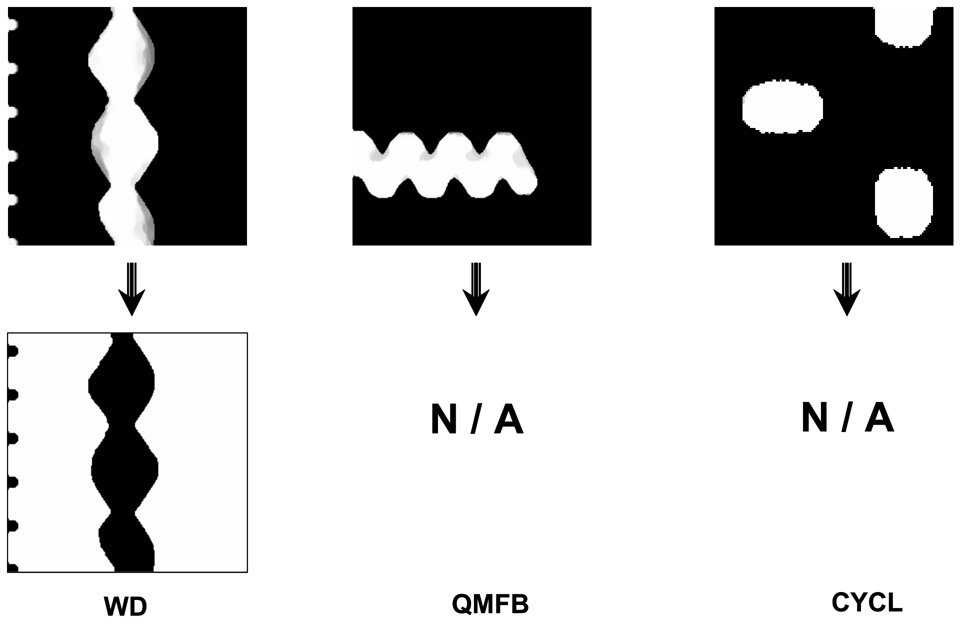


Figure 77 Distance Transform of the F_1_7_500_20_s Signal.

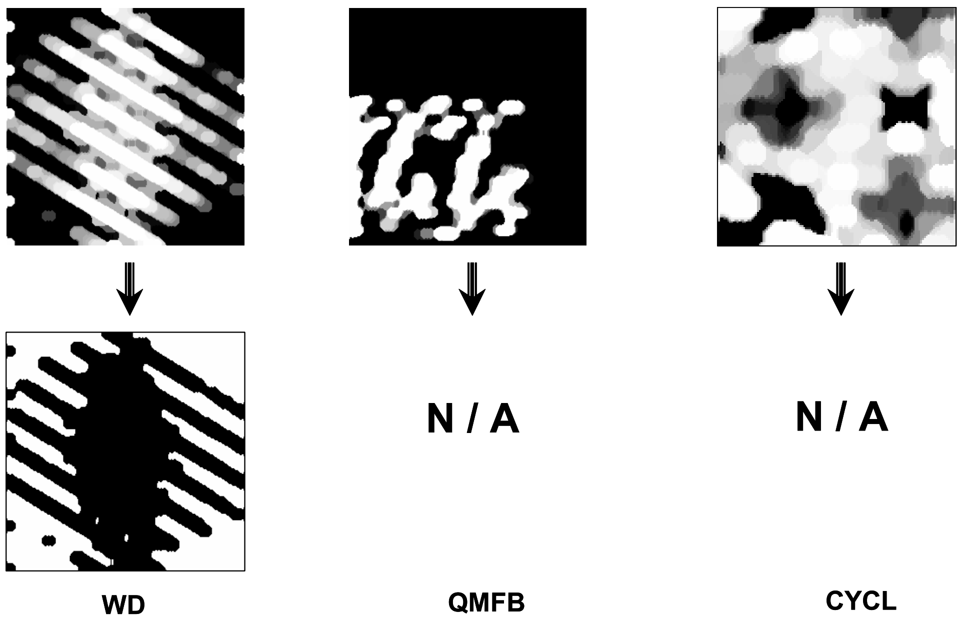


Figure 78 Distance Transform of the FR_1_7_4_1_s Signal.

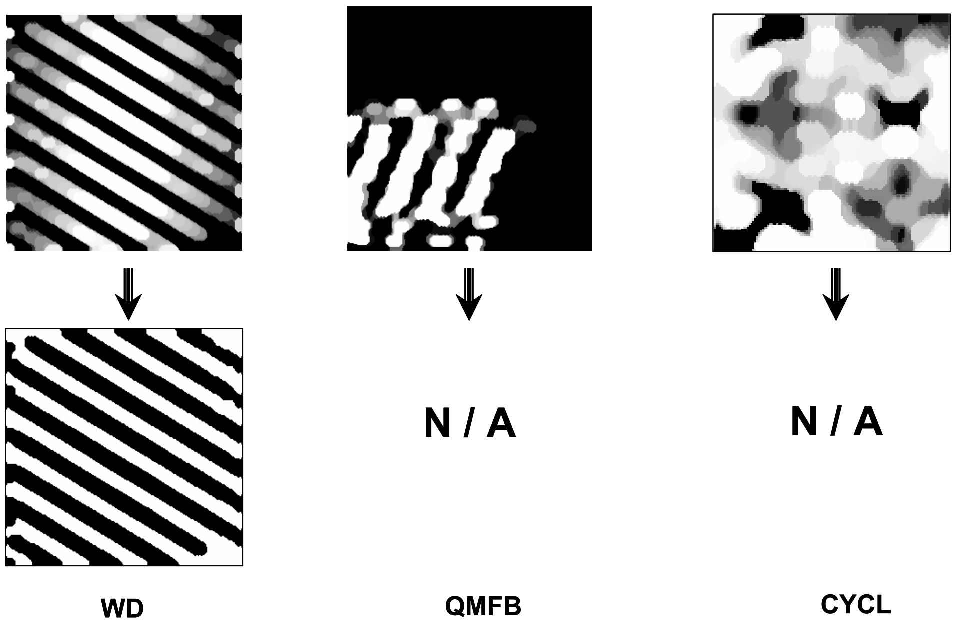


Figure 79 Distance Transform of the P4_1_7_16_1_s Signal.

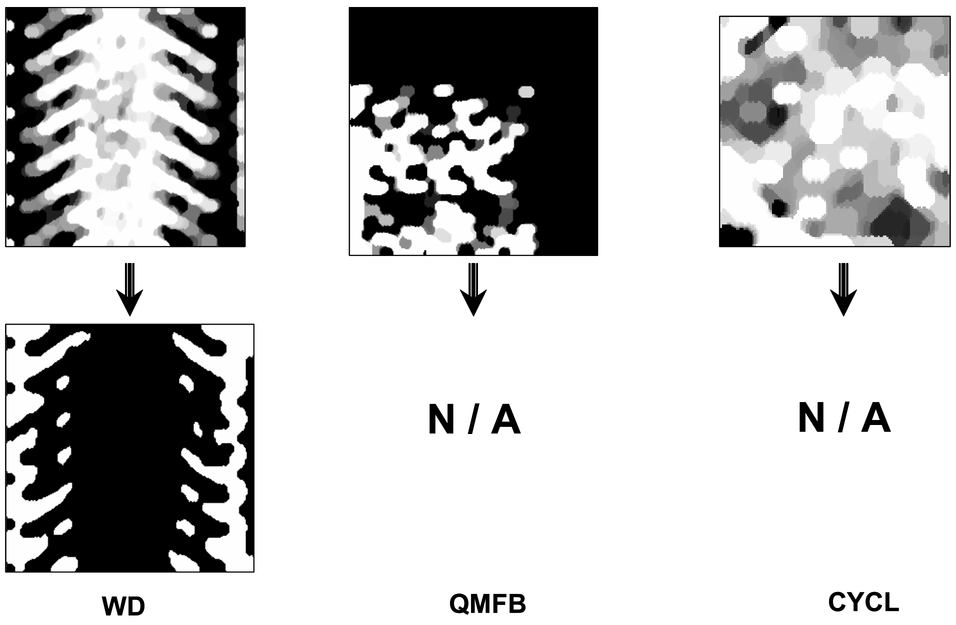


Figure 80 Distance Transform of the PT1_1_7_2_4_s Signal.

f. Image resizing

The final part of the pre–processing is resizing and reshaping the image to a format suitable for the classification process. In this work the resizing was performed in two steps. The first step was to resize the images to a 16×32 matrix using bilinear estimation. The output pixel values were calculated from a weighted average of pixels in the nearest 2-by-2 neighborhoods. The second step was to reshape this reduced image to a 512×1 column vector that will be fed into the classification process. In Figure 81 through Figure 85 the reduction of the images produced by WD, QMFB and CYCL are shown. The different figures show:

- Figure 81 the resizing of the B_1_7_7_7_s signal,
- Figure 82 the resizing of the F_1_7_500_20_s signal,
- Figure 83 the resizing of the FR_1_7_4_1_s signal,
- Figure 84 the resizing of the P4_1_7_16_1_s signal,
- Figure 85, the resizing of the PT1_1_7_2_4_s signal.

All of the resized images clearly display how the contrast of the signals different features is reduced.

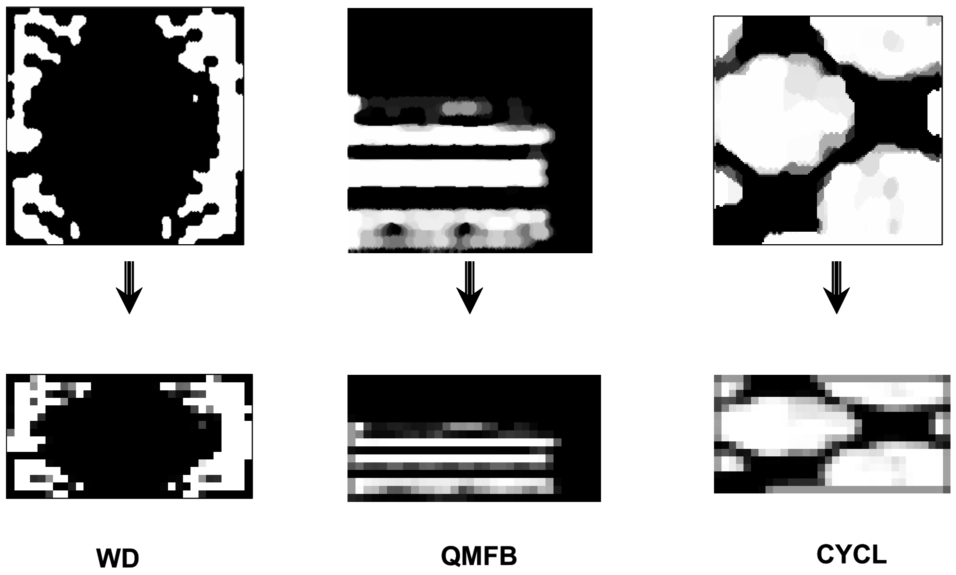


Figure 81 Reshaping of the B_1_7_7_7_s Signal.

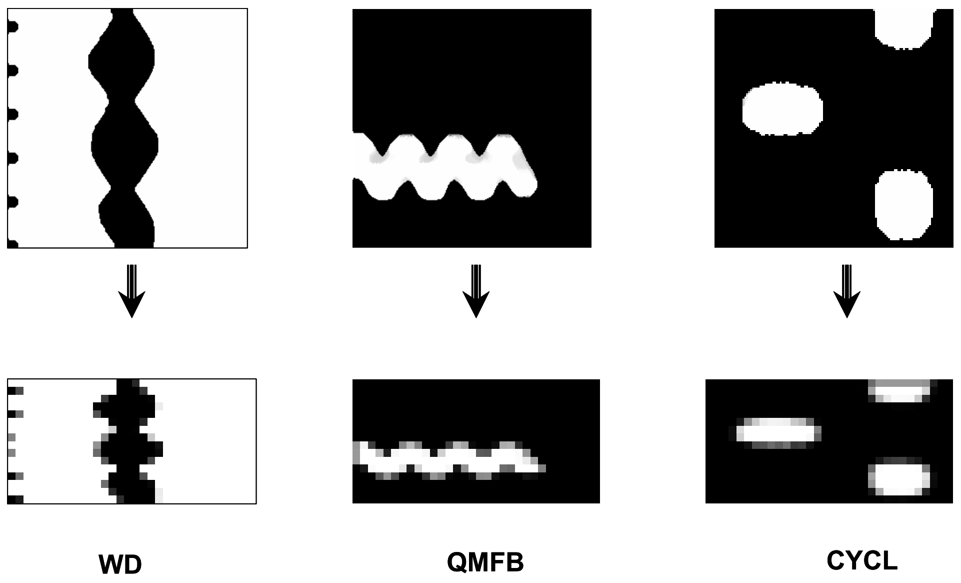


Figure 82 Reshaping of the F_1_7_500_20_s Signal.

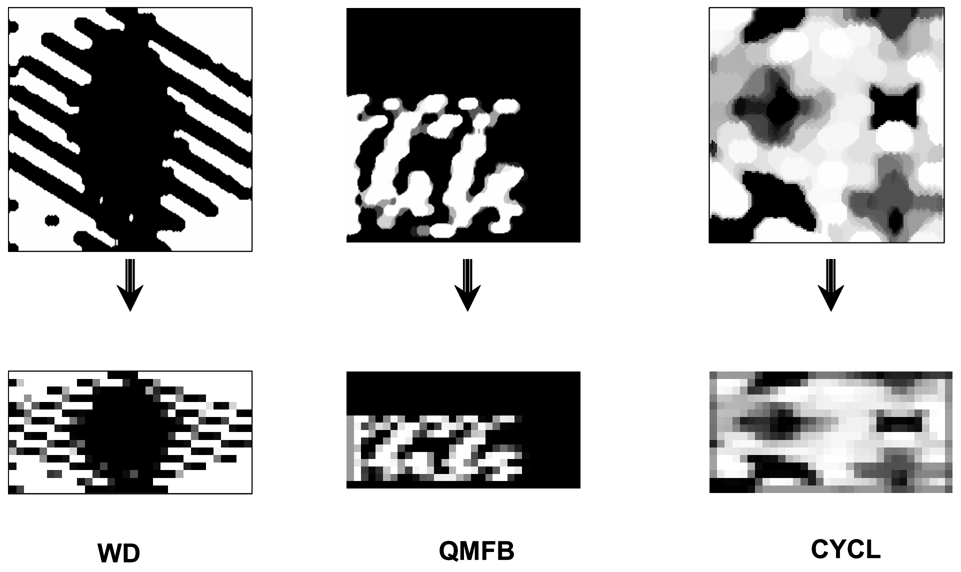


Figure 83 Reshaping of the FR_1_7_4_1_s Signal.

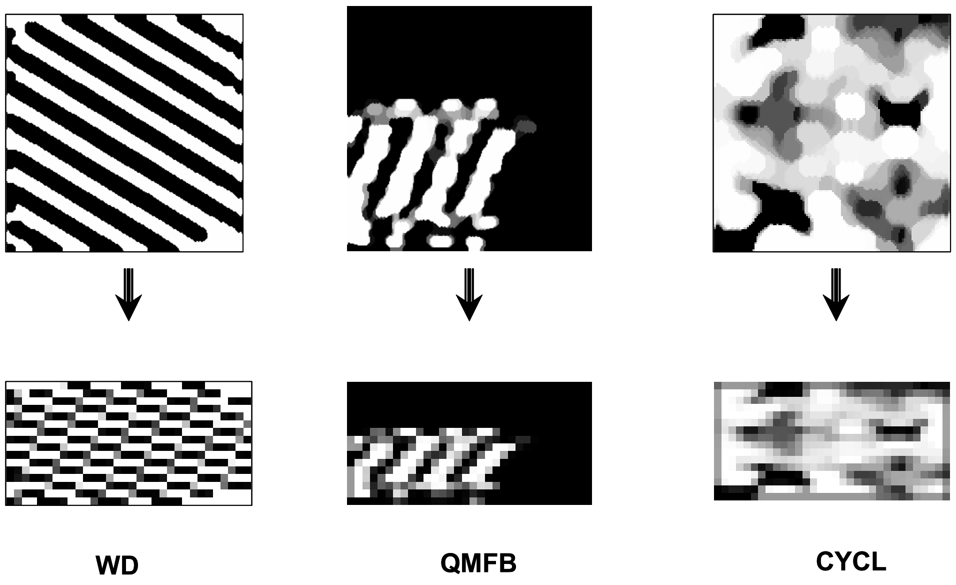


Figure 84 Reshaping of the P4_1_7_16_1_s Signal.

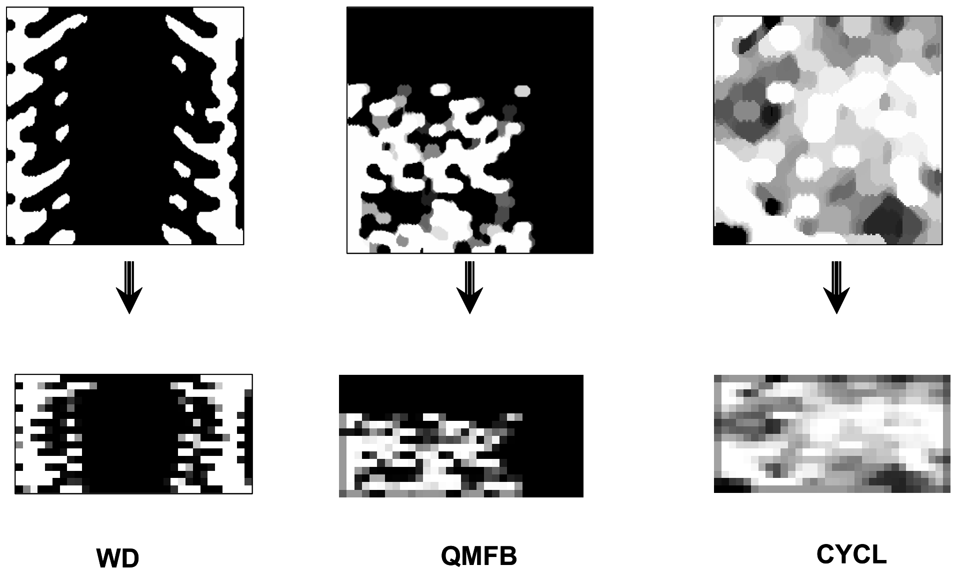


Figure 85 Reshaping of the PT1_1_7_2_4_s Signal.

C. CLASSIFICATION METHOD

1. Classification algorithms

This section will describe the two types of non–linear classifiers, two– and three–layer neural networks used in this work. The section will also describe some different training algorithms for training of the neural networks.

a. Two--Layer Perceptron

A two–layer perceptron shown in Figure 86 is in reality a mapping of the input space x onto the vertices of a hypercube. If the input vector in the l-dimensional space is $\mathbf{x} \in \Re^l$ and p neurons in the hidden layer, then the mapping will be on the vertices of the hypercube in p-dimensional space denoted by H_p were

$$H_p = \left\{ \left[y_1, \ldots, y_p \right]^T \in \Re^p, y_i \in \left[0,1\right], 1 \le i \le p \right\}. \tag{1.13}$$

The vertices of the hypercube are all the points $\left[y_1, \ldots, y_p \right]^T$ of H_p with $y_i \in \left\{0,1\right\}, 1 \le i \le p$.

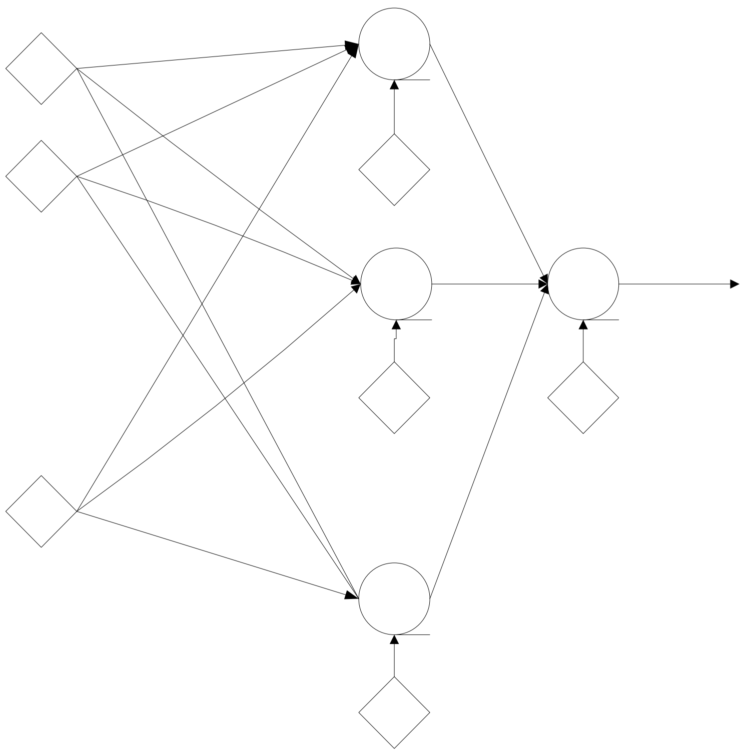


Figure 86 A Two–Layer Perceptron (After [9].).

The mapping of the input space is created by p hyperplanes. Each of those planes is created by a neuron in the hidden layer of the perceptron. An example of the mapping of a simple perceptron with only three neurons in the hidden layer is shown in Figure 87.

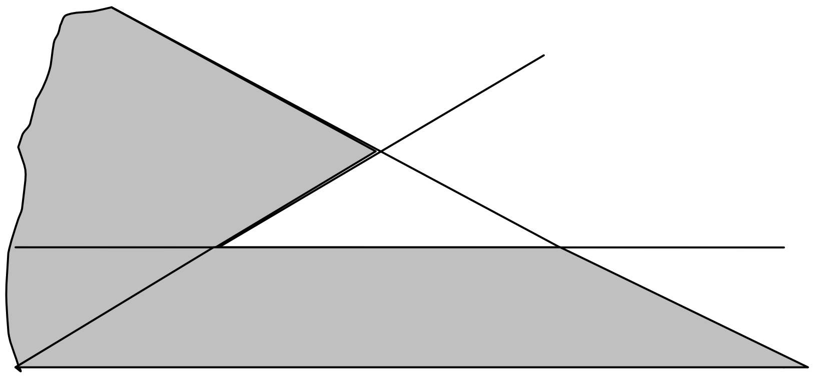


Figure 87 Polyhedra Formed by Neurons in the First Hidden Layer of a Multiplayer Perceptron (After [9].).

Each of the intersections corresponds to a vertex in a three–dimensional cube. It can be said that the first layer of neurons divides the input into polyhedra formed by the intersections of hyperplanes. All feature vectors that are within a region like this are mapped onto a hypercube vertex. The output neuron then creates another hyperplane dividing the hypercube thereby assigning the input to the corresponding output class. A drawback of the two–layer perceptron is that it cannot classify unions of mapped polyhedra. The three–layer neural network structure can solve this problem.

The two–layer perceptron is used in the last of the four neural networks illustrated in Figure 34. It provides the output of the overall classification process. The Neural Network is illustrated in Figure 88.

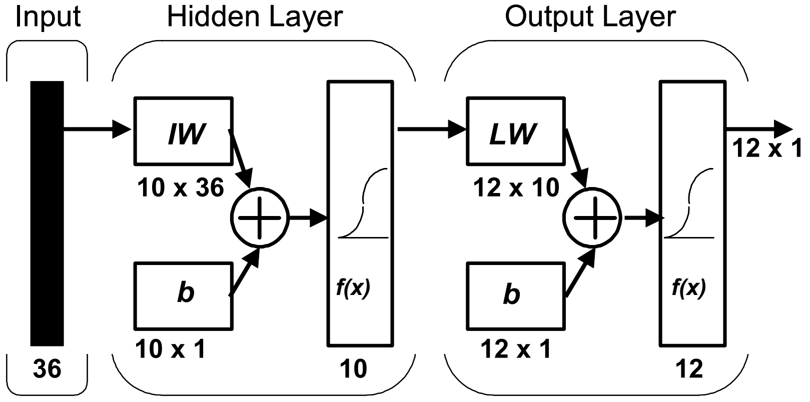


Figure 88 Two–layer Perceptron in the Final Classification Process (After [14].).

The network in Figure 88 has 36 nodes in the input layer corresponding to the number of rows in the three 12-row output vectors produced as a result of the initial classification process. The hidden layer has 10 nodes and the output layer has 12 nodes in which IW, LW and b denote input–weights, layer–weights and bias respectively. The activation function of the network is a sigmoid function, $f(x)$, which is described by

$$f(x) = \frac{1}{1+\exp(-ax)}, \tag{4.14}$$

where a is a slope parameter describing the shape of the decision curve.

b. Three–Layer Perceptron

The two–layer perceptron is unable to separate classes resulting from unions of polyhedra. The reason is that the output neuron can only realize one hyperplane. A way to resolve this problem is to add one hidden layer of neurons. A perceptron formed as that in Figure 89 can resolve any union of polyhedral regions. [9]

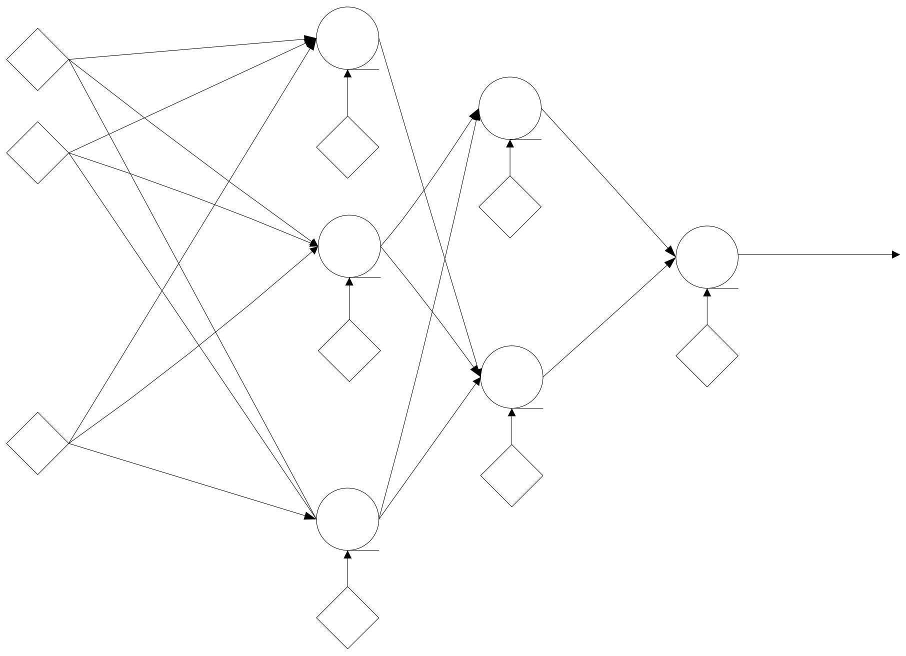


Figure 89 Architecture of a Multilayer Perceptron with Two Hidden Layers and a Single Output Neuron (After [9].).

This is accomplished by the neurons in the first hidden layer forming the hyperplanes, the neurons in the second hidden layer creating regions that describe each class and the output neuron creating the hyperplane that defines the output class. The three–layer perceptron is used in the individual networks in the separate "pipes". Figure 90 illustrates the perceptron.

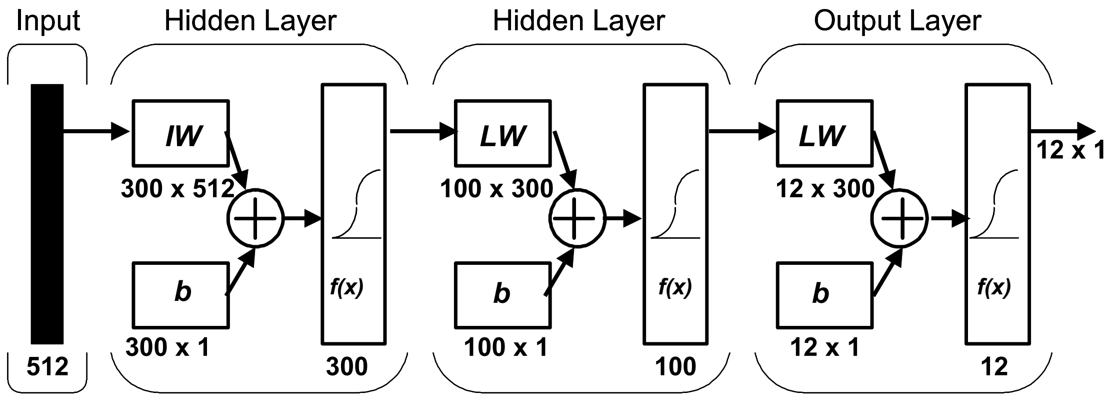


Figure 90 Three–Layer Perceptron Performing Initial Classification (After [14].).

c. Training Algorithms

In order to classify a feature vector correctly all the hyperplanes in the classifier must be correctly described. This assumes that all possible feature vectors are known which is normally not the case. What normally is known is certain points from which these assumptions about the boundaries of the hyperplanes must be made. Today two common ways exist to "train" a classifier to obtain the optimal classification [9]. One is by using a large training set to have the classifier correctly classify all the data. This method is implemented by building the classifier of a series of linear classifiers. The other direction is to use a back propagation algorithm that optimizes the weights in the classifier by computation of a "cost" for errors in the classification.

In this work the back-propagation algorithm was used. The steps in this algorithm are:

- Initialization: All weights are initialized to a small random value.
- Forward computations: For each training feature vector, $\mathbf{x}(i)$, where $i = 1, 2, \ldots, N$ and N is number of training pairs, all $\upsilon_j^{r(i)}$, $y_j^r(i) = f\left(\upsilon_j^r(i)\right)$, $j = 1, 2, \ldots, N$, $r = 1, 2, \ldots N$ is calculated. Here $\upsilon_j^{r(i)}$ is the weighted summation of the inputs to the j-th neuron of the r-th layer. The variable $y_j^r(i)$ is the corresponding output after the activation function. After this the cost-function $J = \sum_{i=1}^{N} \varepsilon(i)$ is calculated. Here $\varepsilon(i)$ is a function that takes on a certain value for each input/output pair. In this work this function is the sum of squared errors in the output neuron.
- Backward computations: For $i = 1, 2, \ldots, N$ and $j = 1, 2, \ldots, N$, computation of $\delta_j^r(i)$ which is defined as $\delta_j^r(i) = \dfrac{\partial \varepsilon(i)}{\partial \upsilon_j^r(i)}$ is made.
- Update of the weights: The weights are updated using

$$\mathbf{w}_j^r(new) = \mathbf{w}_j^r(old) + \Delta \mathbf{w}_j^r,$$

where

$$\Delta \mathbf{w}_j^r = -\mu \sum_{i=1}^{N} \delta_j^r(i)\,\mathbf{y}^{r-1}(i).$$

The variable μ is the learning constant that determines the convergence speed of the algorithm.

Training of the neural networks was first done using only the pure signals (no noise). Next the noisy signals were introduced and finally the networks were trained with the pure signal again. The reason is to give the networks an initial setting of the weights close to the optimum value before the noisy signals are introduced. The final training with the pure signal is to reset eventual erroneous weights introduced by the training with noise. The network is supposed to perform 100% with the training signals. The maximum number of training iterations was set to 15000 times.

The results of the training algorithm for the four neural networks are shown in Figure 91 to Figure 94 where plot (a) in each figure describes the initial noise-less training session and plot (b) describes the training with noise. Figure 91 shows the training results for the neural network classifying the WD representation. Figure 92 shows the results for the neural network classifying QMFB representations. Figure 93 shows the CYCL result and Figure 94 shows the result for training of the combined neural network. The figures illustrate the difficulty the neural network has in mapping a noisy signal. It takes substantially more training iterations (epochs) during training with a noisy signal before the weights converge.

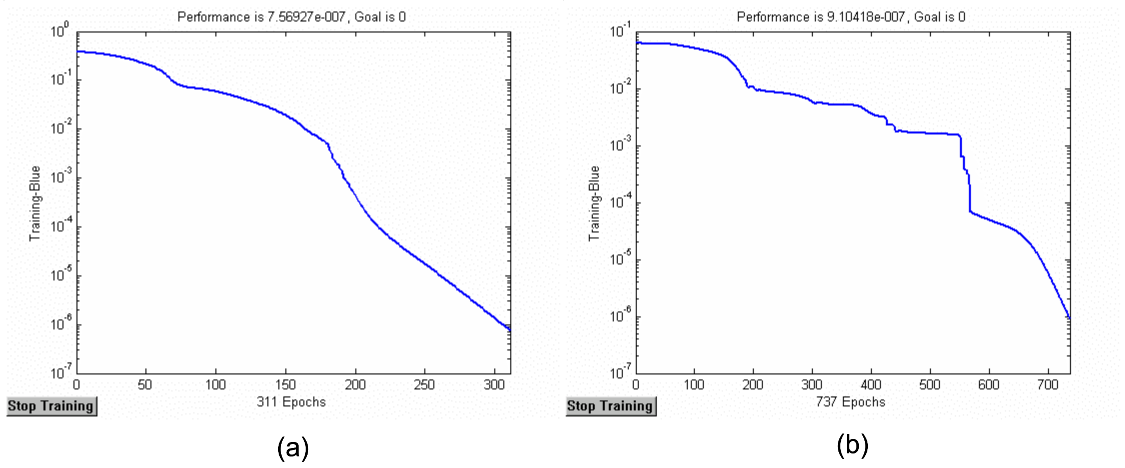
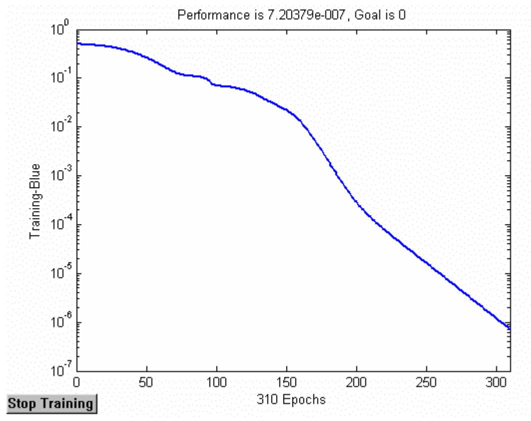
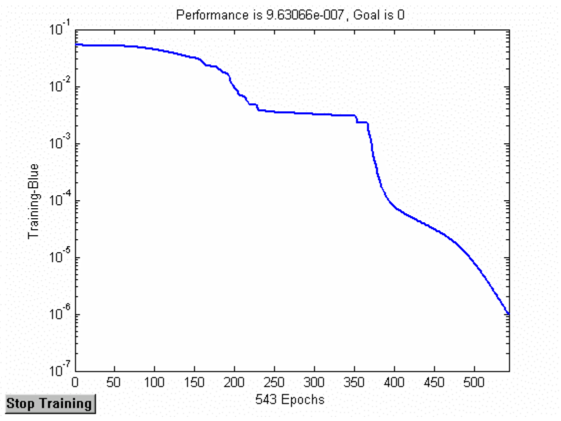


Figure 91 Training of the Neural Network Classifying Images from the Wigner–Ville Distribution.

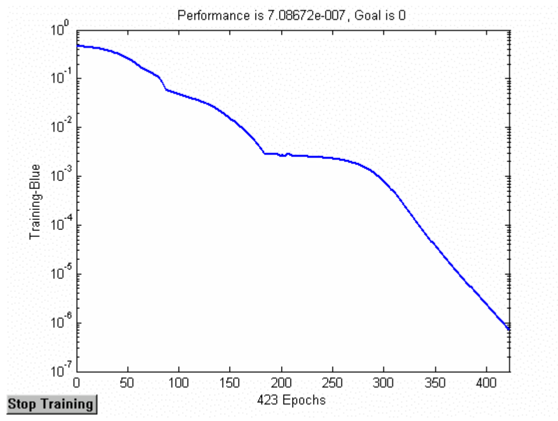


(a)

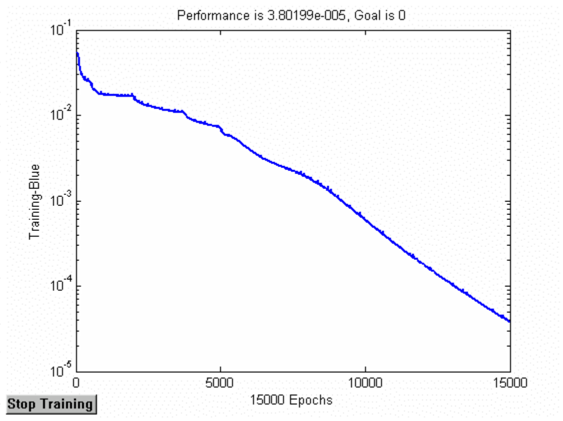


(b)

Figure 92 Training of the Neural Network Classifying Images from the Quadrature Mirror Filter Bank.

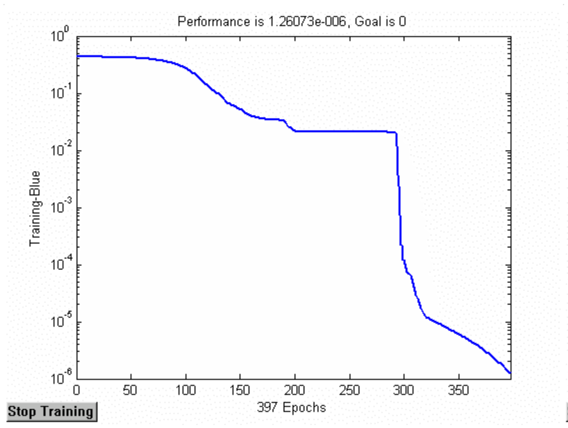


(a)

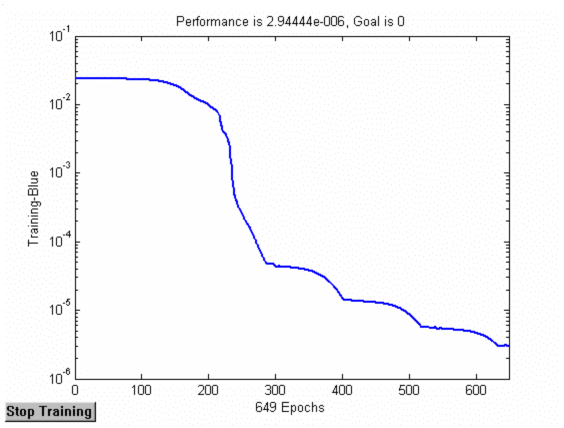


(b)

Figure 93 Training of the Neural Network Classifying Images from the Cyclo–Stationary Processing.



(a)



(b)

Figure 94 Training of the Neural Network Giving the Final Classification Result.

2. Classification Result

The two sets of training signals previously generated were used to test the classifier. The result of the test signals being passed through the networks are shown in Figure 95 and Figure 96 where Figure 95 shows the result produced by the classification algorithm being tested with the modulation varied signals found in Table 4. Figure 96 shows the result when the classification algorithm is tested with signals using varied noise levels. Those test signals are shown in Table 5. To make the result more readable, Table 10 shows the result in numbers.

In the diagrams, the numbers on the "Signal Type" axis corresponds to a certain signal being introduced to the classification algorithm. The numbers on the Modulation type axis corresponds to the numbers shown in Table 7. The correct signals are plotted in the diagram using the symbol denoted "True Value." The result from the fourth, combining neural network is plotted in the diagrams using the symbol denoted "Result All." The results from the classification processes in the three separate "pipes" are plotted using the symbols denoted "Result WD," "Result CYCL" and "Result QMFB." The first tested signal can serve as an example of how to read the table. The star shows that the true value is 1 (BPSK modulation). The result from the combined network, denoted by the diamond, is 8 (P4). The WD neural network, denoted by a square, also gives 8 (P4) as output while the CYCL neural network, denoted by triangle, predicts a 12 (T4). Finally, the QMFB neural network, denoted with an x, gives 3 (FMCW) as a result.

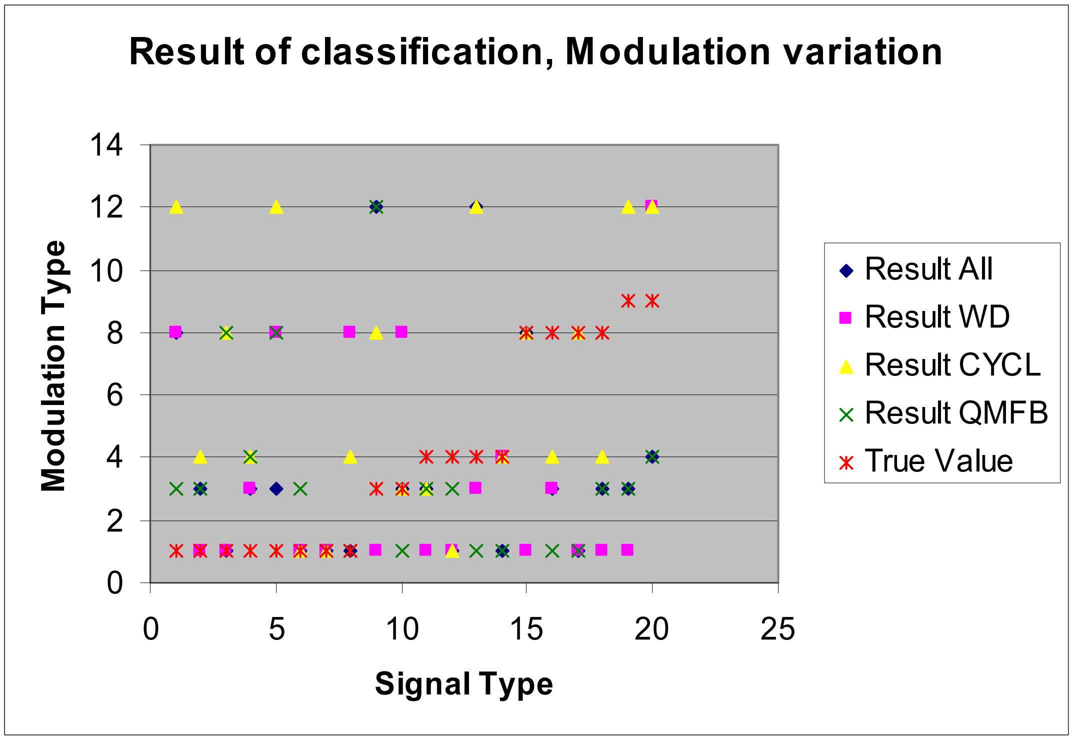


Figure 95 Classification Results for Modulation Variation.

Figure 95 shows the result when the algorithm is being tested by various changes in the modulation types. The result is disappointing because only 11 out of 20 signals are being classified correctly by any of the neural networks. The fourth, combining neural network, which classifies 6 out 20 correctly, does not differ in performance compared with the three other networks. Table 10 shows the overall result of the classification. As described for Figure 95, the performance of the combined network is not better than the three separate networks.

Table 10 Result of Classification

CLASSIFICATION RESULT	
Modulation variation (20 test signals)	
Detection Method	# correct classifications
Combined	6
WD	5
CYCL	6
QMFB	3
Noise variation (90 test signals)	
Detection Method	# correct classifications
Combined	42
WD	46
CYCL	26
QMFB	43

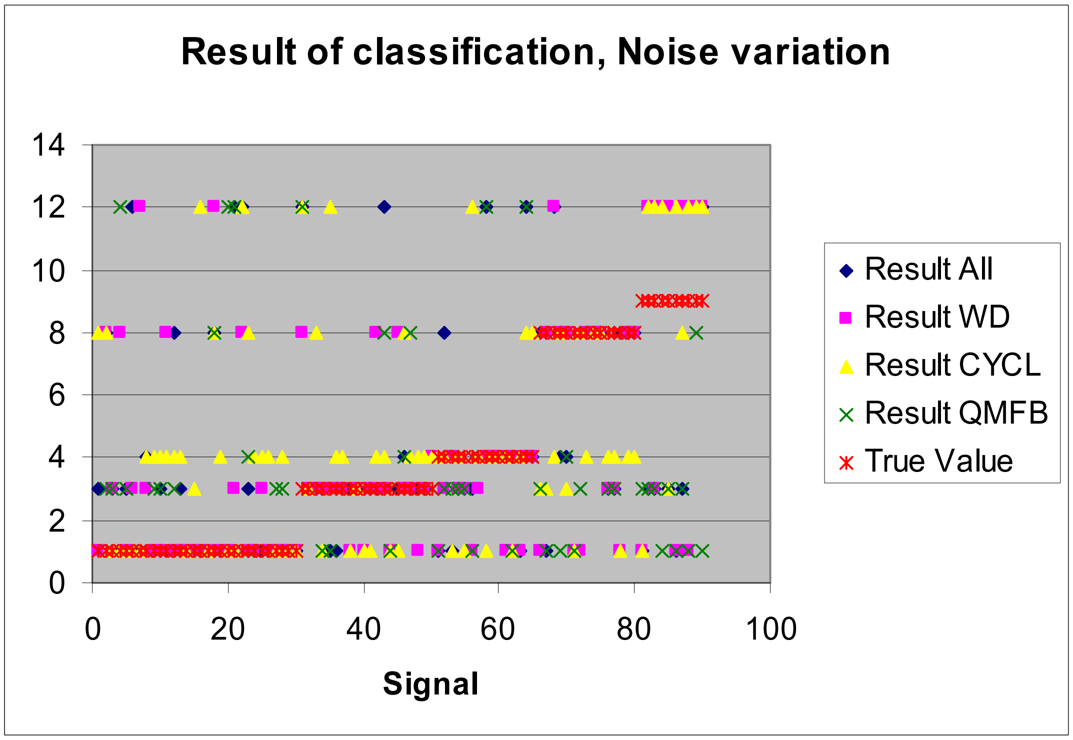


Figure 96 Classification Results for Noise Level Variation.

The result of testing the algorithm with noise–varied signals is slightly different. The result from Figure 96 and Table 10 shows that the combining network correctly identifies 42 of 90 signals. The Wigner–Ville distribution correctly identifies 46 out of 90 indicating that the pre–processing of this image representation has been more successful than the others. It might also indicate that the WD representation gives a more distinct representation for each modulation type than the representations produced by QMFB and CYCL. In general the second set of results indicates an overall better performance of the algorithm when only the variation level is varied between the signals.

A surprising result in the second test is that the QMFB classification performs worse than the other detection methods. Since this detection method gives a very distinct representation of a pure signal, the result indicates that QMFB is sensitive to noise.

The result of the second test further indicates that the CYCL representation is sensitive to noise. A comparison of the CYCL images for a P1 signal and a Frank–coded signal in Figure 97 verifies this. The figure shows two different noise levels for each of the signals causing the difference between the two modulation types hard to detect. Column (a) shows a Frank code signal with 0 dB (top) and 3 dB (bottom) signal–to–noise ratio (SNR). Column (b) shows a P1 signal with 0 dB (top) and 3 dB (bottom) SNR. This makes it hard for a classification algorithm based on image processing to correctly classify these signals.

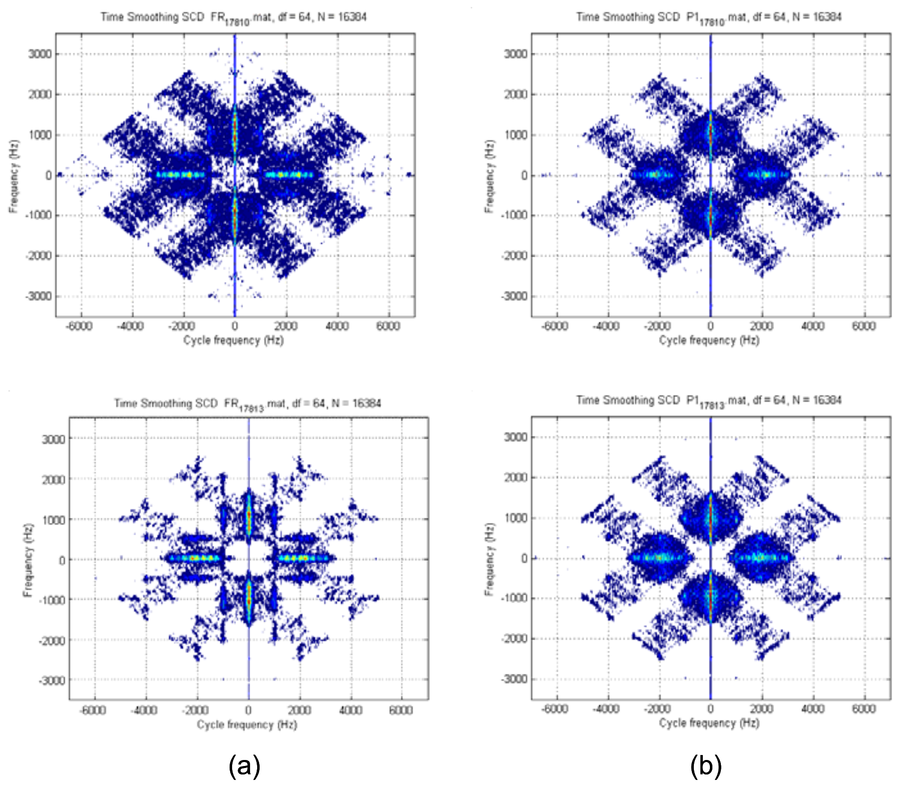


Figure 97 Image Representation of a FR_1_7_8_1 (a) Signal and a P1_1_7_8_1 (b) Signal with 0 dB and 3 dB SNR.

Another factor that might affect the classification result is the four networks that are trained to handle 12-row output vectors. That is, the neural networks can give an output to nodes that have been trained to give only zero output, thereby increasing the misclassification rate. A test was performed using 5 rows in the output vectors with the result being shown in Table 11. The combined network shows a significant improvement when classifying the test signals with modulation variation. Additionally, in the case of noise variation, the result is improved. This indicates that it is important to train the networks for the correct output.

Table 11 Classification Result, 5-Row Output Vectors

CLASSIFICATION RESULT	
Modulation variation (20 test signals)	
Detection Method	# correct classifications
Combined	9
WD	7
CYCL	8
QMFB	2
Noise variation (90 test signals)	
Detection Method	# correct classifications
Combined	48
WD	54
CYCL	41
QMFB	47

Table 11 shows that the Wigner–Ville distribution outperforms the combined classification made by the fourth neural network. This is similar to the worse result produced by [15] when comparing different feature extraction method. A difference between the result presented in [15] and the result in this work is the big variation of the image representation produced by the detection methods in this work shows when noise is introduced or the modulation is slightly changed. It is important that the detection methods used to produce the images fed into the combined classification algorithm has a distinct representation for each modulation type and each variation of the modulation type. The variation can be either with the addition of noise or by a slight change in the modulation.

D. SUMMARY

This chapter has shown the proposed approach to pre-processing, feature extraction and classification. The concepts of pre–processing were illustrated with actual images from the processed signals and the implemented classification process was described. The results from the developed classification process were presented at the end of the chapter.

The result using the algorithm is worse than expected in that the combined classification algorithm only classified approximately 53% correctly. The individual classification based on the Wigner–Ville distribution performed better than the classification methods based on CYCL or QMFB with 64% of the test signals correctly classified.

The next step when a signal is classified is to extract the significant parameters. Chapter VI gives a brief description of a proposed method to do the extraction of the parameters using image processing.

V. EXTRACTION OF SIGNIFICANT PARAMETERS

Classification of a signal is only the first step in identifying a radar emission. After the radar signal is classified, extraction of key parameters from the signal is performed. Those parameters are then combined to serve as the basis of deciding upon the type of radar station it is, the platform the radar support and, if applicable, the weapon systems associated with this radar station.

This chapter gives a brief description of a proposed method to extract significant parameters from a signal once it is classified. The description is only conceptual with a few images as examples, since limited time was spent on this part of the thesis work.

A. PROPOSED METHOD OF PARAMETER EXTRACTION

The method proposed to extract key parameters from the radar signal is shown in Figure 98. The method starts where the classification method described in the previous chapter ends. Once the classification decision of the signal is made, a table lookup is performed. The table used contains information showing which detection method most accurately describes the characteristics of the now classified signal. The output from this detection method is then used for parameter extraction. In some cases a recalculation of the detection algorithm may be necessary to acquire the most accurate and distinct representation of the classified signal. An example of the latter is recalculation of a different layer in the quadrature mirror filter bank compared to the layer used in the classification process.

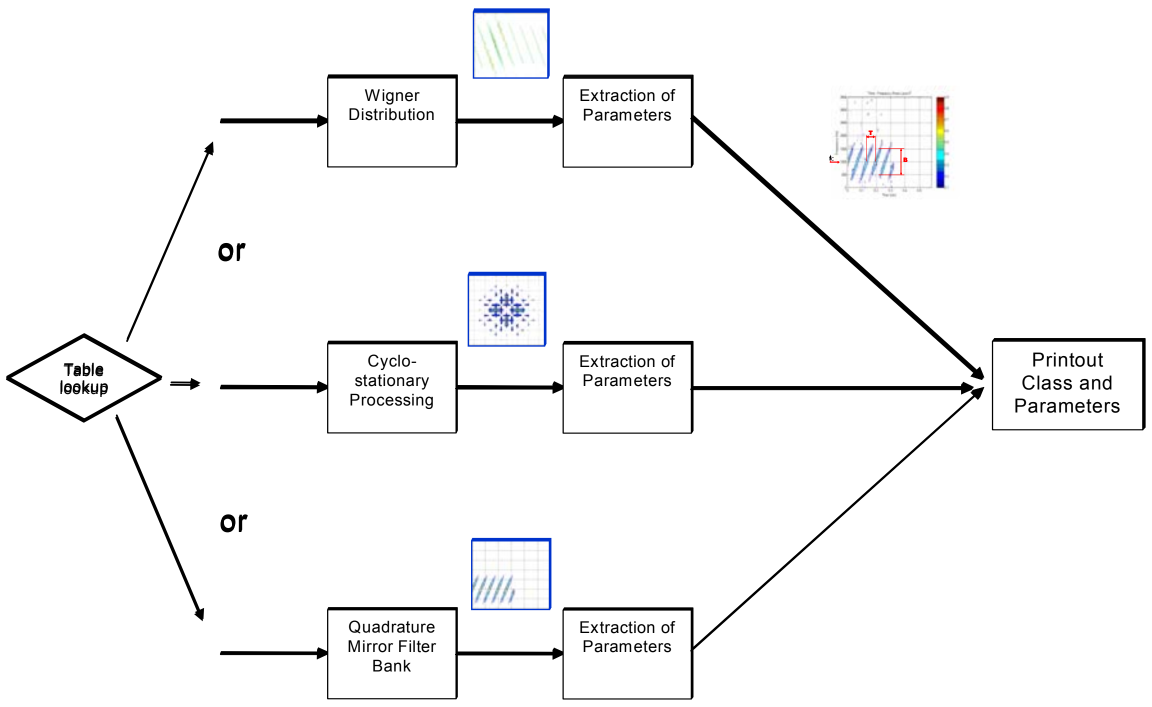


Figure 98 Illustration of Proposed Method to Extract Key Parameters from a Classified LPI Signal.

The QMFB normally generates the clearest representation of a signal and is especially suitable for the type of processing used to determine the signals parameters. Figure 99 shows how the significant parameters are proposed to be extracted from a Frank–code signal with $f_c = 1000$ Hz, $f_s = 7000$ Hz, $SNR = 3$ dB, $M = 8$ frequency steps/samples per frequency and 1 cycle per period using quadrature mirror filter bank.

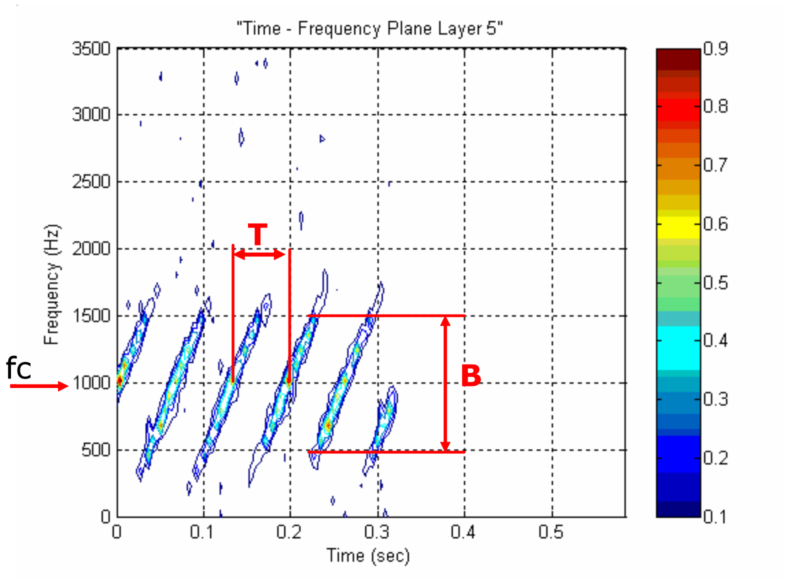


Figure 99 Illustration of how the Key Parameters are Extracted for a Frank–Coded Signal using QMFB.

By looking at the projection of the signal on the frequency axis (using summation along the rows in the image matrix), it is possible to determine the bandwidth, B, by using a 50% threshold on the signal displayed on the frequency axis. Measuring the width of the resulting image then gives the bandwidth. Once the bandwidth is determined, the center frequency, f_c. can be determined by taking the center of the bandwidth. The period, T, is calculated by projecting 20 rows on each side of the center-frequency onto the time axis. The period is then formed by measuring of the distance between the peaks. Figure 100 shows the projection of the signal (without gridlines) onto the frequency axis. The bandwidth is relatively easy to determine and by taking the center of the bandwidth it is possible to obtain a decent value of the carrier frequency. Figure 101 illustrates the projection on the time–axis.

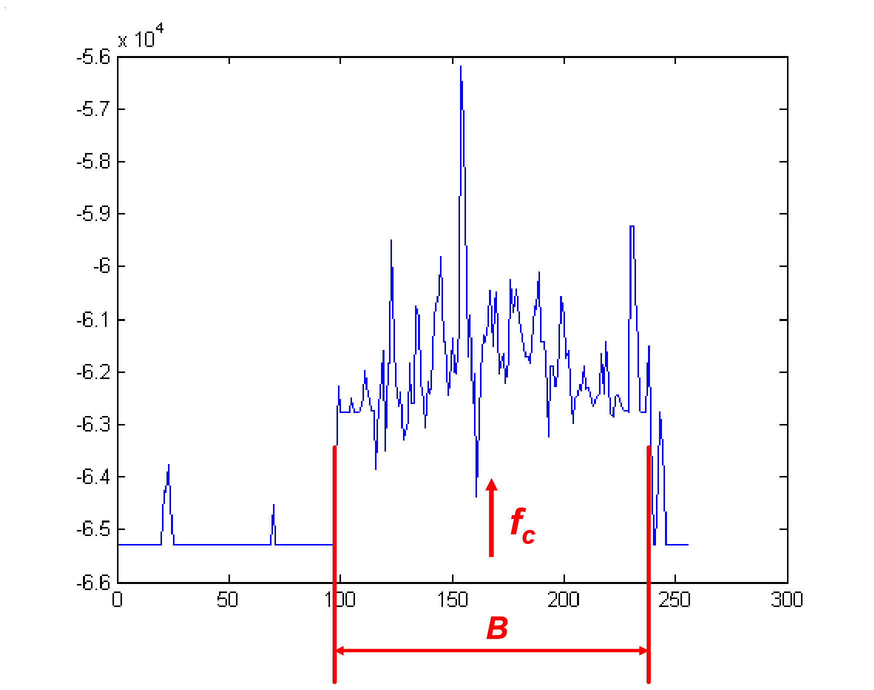


Figure 100 Projection of the Image Representing the Frank–Coded Signal onto the Frequency Axis.

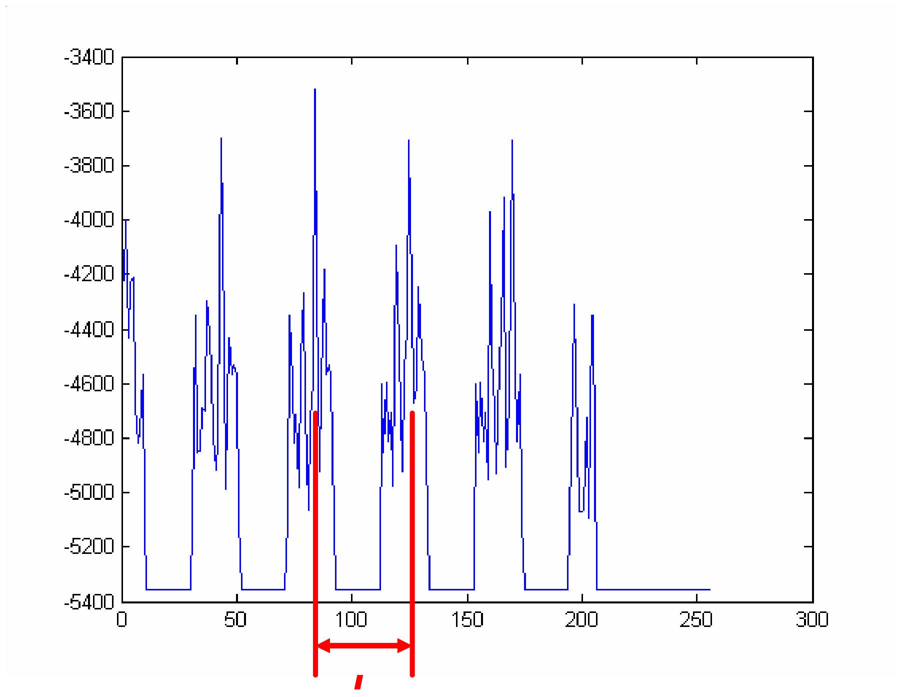


Figure 101 Projection of 10 Rows Around f_c onto the Time Axis.

Using these three significant parameters, the calculation of several other parameters is possible, depending on the type of signal. It is also possible to make other measurements by performing further image processing. Further information can be found in Refs. [1, 6].

B. SUMMARY

Measuring the significant parameters using the image representation is possible from one of the three detection methods. By initially determining the significant parameters, several other parameters describing a signal can be calculated. A prerequisite is that the signal is known and classified to be able to use the correct image representation of the signal and to apply the most appropriate image processing method.

This chapter concludes the description of this work. In Chapter VII the concluding remarks and recommendations for further work are given.

THIS PAGE INTENTIONALLY LEFT BLANK

VI. CONCLUSIONS AND RECOMMENDATIONS

A. CONCLUSIONS

In this thesis image representation of LPI radar signals generated by the Wigner distribution, cyclostationary processing and quadrature mirror filter bank were used to determine what type of modulation was present, using three separate neural network parallel classification algorithms. The final classification result was generated by combining the separate classification results using a fourth separate neural network.

The best result achieved was a correct classification of approximately 53%. The expected outcome was somewhat higher. The difference in the image output from the detection methods when either noise was introduced to the signal or the modulation was changed proved to be more problematic for the cyclostationary processing than the other detection methods. The classification using the Wigner–Ville distribution performed better than the classification using the cyclostationary processing or the quadrature mirror filter bank. Also the combined classification with inputs from all three detection methods was outperformed by the classification based on Wigner–Ville distribution only. An explanation to this might be that the cross–terms being present in the image help creating a distinct representation for each signal.

The work first concludes that the feature extraction for each image needs to be improved in order to find more distinct features and, thereby, increase the classification result. Secondly, the detection methods used in the classification process must represent a signal in a distinct manner with noticeable differences between slight variations in noise and modulation. The neural networks used in the classifiers provides some compensation for variations in the inputs to the classification process but, in order to perform well, the networks need to be trained sufficiently.

B. RECOMMENDATIONS

Recommendations for further work are initially to investigate other feature extraction methods (statistical, automatic selection, etc.) that might improve the classification result. When a suitable classification of the test sets are done, investigate the effect if more signals are added to the inputs. It is also recommended to test the effect on the classification result if the fourth, combining neural network is excluded from the algorithm and the final decision is made in the form of a simple majority decision based on the output vectors from the three separate neural networks. Tests should further be made by the use of real, sampled signals to detect any differences between the models and the reality.

When a suitable pre–processing/feature extraction is developed and the classification result is increased, a hardware version for high speed, near real–time, computation should be developed to test the algorithm in a real environment.

APPENDIX A

Code developed in MATLAB simulating the polytime phasecodes T1(n) through T4(n).

```
%==========================================================================
%POLYTIME CODE T1-T4
%By Christer Persson
%October 28, 2002
%==========================================================================

clear all;
clc;
disp('*********************************************');
disp('***********POLYTIME CODE (T1-T4)************');
disp('*********************************************');

%-----------------------------------------------------------------------
%DEFAULT VARIABLES
%-----------------------------------------------------------------------

A=1;                    % Amplitude of CW
f=1e3;                  % Carrier frequency
fs=7e3;                 % Sample Rate
SNR_dB = 0;                 % Signal to Noise Ratio
scale=30;                 % Scaling for plotting time domain graphs
m=2;                    % Number of phase states
SAR=ceil(fs/f);             % Sampling rate
k=4;                   % Number of stepped frequency segments
T=16e-3;                  % Overall codeperiod
Ts=1/(fs);                % Sampling Period
deltaphi=2*pi/m;             % fundamental stepsize
deltaf=250;                 % Modulation bandwidth

%-----------------------------------------------------------------------
% NEW INPUT IF NEEDED
%-----------------------------------------------------------------------

newvar = 1;
while newvar == 1;
   disp(' ')
   disp('WHICH PARAMETER DO YOU WANT TO SET ?  ')
   disp(' ')
   fprintf('1. Amplitude of the carrier signal - A= %g.\n', A)
   fprintf('2. Carrier frequency - f (Hz) = %g.\n', f)
   fprintf('3. Sampling frequency - fs (Hz)= %g.\n', fs)
   fprintf('4. Signal to noise ratio - SNR_dB (dB) = %g.\n', SNR_dB)
   fprintf('5. Number of phase states - n = %g.\n', m)
   fprintf('6. Number of segments - k = %g.\n', k)
   fprintf('7. Modulation Bandwidth - df (Hz)= %g.\n', deltaf)
   fprintf('8. Overall Code period - T (s)= %g.\n', T)
   fprintf('9. No changes\n')
```

```
    disp(' ')
    option= input('Select a option: ');

    switch option
        case 1
            A=input('New amplitude of the carrier signal= ');
        case 2
            f=input('New carrier frequency (Hz) = ');
        case 3
            fs=input('New sampling frequency (Hz)= ');
        case 4
            SNR_dB=input('New signal to noise ratio (dB)= ');
        case 5
            m=input('New number of phase states =');
        case 6
            k=input('New number of segments =');
        case 7
            deltaf=input('New Modulation Bandwidth =');
        case 8
            T=input('New Overall Code period =');
        case 9
            newvar = 0;
    end
    clc;
end

%=========================================================================
% Phase code for T1-T4 from IEEE Transactions on Aerospace and Electronic
% systems Vol. 35, NO 2 April 1999.
% POLYTIME CODING AS A MEAN OF PULSE COMPRESSION
% by John E. Fielding.
%=========================================================================
newvar2 = 1;
while newvar2 == 1;
    disp(' ')
    disp('WHICH PHASECODE DO YOU WANT TO USE ?  ')
    disp(' ')
    disp('1. T1')
    disp('2. T2')
    disp('3. T3')
    disp('4. T4')
    disp(' ')
    option2 = input('Select a phasecode ');
    switch option2

        case 1

            % Compute the T1 Polytime Phase (formula from paper)
            ttt=1;
            for tt = 0:(Ts):(T-Ts)
                jj = floor(k*tt/T);
                phase(ttt)= mod(((2*pi/m)*floor(((k*tt - jj*T)*(jj*m/T)))), 2*pi);
                if ttt==1
                    phaseunwrapped(ttt)=phase(ttt);
                else
                    if phase(ttt)==phase(ttt-1)
            phaseunwrapped(ttt)=phaseunwrapped(ttt-1);
        else
            phaseunwrapped(ttt)=phaseunwrapped(ttt-1)+2*pi/m;
        end
    end
    ttt= ttt+1;
end
newvar2=0;

case 2

    %Compute the T2 Polytime Phase (formula from paper)
    ttt=1;
    for tt = 0:(Ts):(T-Ts)
        jj = floor(k*tt/T);
        phase(ttt)= mod(((2*pi/m)*floor((((k*tt - jj*T)*((2*jj-k+1)/T)*(m/2))))), 2*pi);
        if ttt==1
            phaseunwrapped(ttt)=phase(ttt);
        else
            if phase(ttt)==phase(ttt-1)
                phaseunwrapped(ttt)=phaseunwrapped(ttt-1);
            else
                phaseunwrapped(ttt)=phaseunwrapped(ttt-1)+2*pi/m;
            end
        end
        ttt= ttt+1;
    end
    newvar2=0;

case 3

    %Compute the T3 Polytime Phase (formula from paper)
    ttt=1;
    for tt = 0:(Ts):(T-Ts)
        phase(ttt)=mod(((2*pi/m)*floor((m*deltaf*tt.^2)/(2*T))),2*pi);
        if ttt==1
            phaseunwrapped(ttt)=phase(ttt);
        else
            if phase(ttt)==phase(ttt-1)
                phaseunwrapped(ttt)=phaseunwrapped(ttt-1);
            else
                phaseunwrapped(ttt)=phaseunwrapped(ttt-1)+2*pi/m;
            end
        end
        ttt = ttt+1;
    end
    newvar2=0;

case 4

    %Compute the T4 Polytime Phase (formula from paper)
    ttt=1;
    for tt = 0:(Ts):(T-Ts)
        phase(ttt)=mod(((2*pi/m)*floor((m*deltaf*tt.^2)/(2*T)-(m*f*tt)/2)),2*pi);
        if ttt==1
            phaseunwrapped(ttt)=phase(ttt);
            else
               if phase(ttt)==phase(ttt-1)
                  phaseunwrapped(ttt)=phaseunwrapped(ttt-1);
               else
                  phaseunwrapped(ttt)=phaseunwrapped(ttt-1)+2*pi/m;
               end
            end
            ttt = ttt+1;
         end
         newvar2=0;
   end
   clc;

end

Phase=deltaphi*floor(phase/deltaphi);

%-----------------------------------------------------------------------
% This section generates I & Q without T1-T4 phase shift and I & Q with Phase
% shift.  The signals are generated five times by the outer loop.  The
% variable 'index' is used to generate a time vector for time domain plots.
%-----------------------------------------------------------------------

index=1;
for p=1:5 %Generate the signal five times and store sequentially in corresponding vectors

   for loop=1:(ttt-1) %Loop to shift phase

      I(index)=A*cos(2*pi*f*(loop-1)*Ts+Phase(loop)); %Calculate in phase component of signal with
phase shift
      IWO(index)=A*cos(2*pi*f*(loop-1)*Ts); % Calculate in phase component of signal without phase
shift
      Q(index)=A*sin(2*pi*f*(loop-1)*Ts+Phase(loop)); % Calculate quadrature component of signal with
phase shift
      QWO(index)=A*sin(2*pi*(loop-1)*Ts); %Calculate quadrature component of signal without phase
shift
      time(index)=(index-1)*Ts; %time vector cumulation
      index = index+1;

   end

end

%Power Spectral Density for I with phase shift & with WGN with Signal to noise ratios (SNR) = [0,-
5,5,10,-10,-20]
%for loop makes calculations and plots for each value of SNR for WGN
[a,b]=size(I);
SNR=10^(SNR_dB/10);
power=10*log10(A^2/(2*SNR));%calculate SNR in dB for WGN function
noise=wgn(a,b,power);%calculate noise at specified SNR
IN=I+noise;              %add noise to I with P4 phase shift
IPWON=I;                %I with phase shift without noise
QN=Q+noise;             %add noise to Q with P4 phase shift
QPWON=Q;               %Q with phase shift without noise
```

```
%****************************************************
%PLOTS
%****************************************************

disp(' ')
plt = input('Do you want to generate plots of the signal (Y/y or N/n) ?','s');
disp(' ')
if (plt == 'Y') | (plt =='y')
   disp(' ')


   %Plot Power Spectral Density for I without phase shift
   figurecount=1; %figurecount is plot index
   figure (figurecount); % open new figure for plot
   psd(IWO,[],fs); %Power Spectral Density of I without Phase shift
   title(['Fig # ' num2str(figurecount) ', PSD of I without Phase Shift or Noise']);
   figurecount=figurecount+1; %increment figure count


   %time domain plot of in phase signal I without phase shift
   figure (figurecount); % open new figure for plot
   % plot small portion of time domain signal I so that data will fit meaningfully in figure.
   %floor(size(time,2)/scale) selects a small sample of the vectors to plot
   plot (time,IWO);
   title(['Fig # ' num2str(figurecount) ', Time Domain of I without Phase Shift or Noise']);
   xlabel('{\itTime - Seconds} ');
   ylabel('Amplitude');
   grid on;
   figurecount=figurecount+1; %increment figure index


   %Power Spectral Density for I with phase shift
   figure (figurecount); %open new figure for plot
   psd(I,[],fs); %plot power spectral density of I with phase shift
   title(['Fig # ' num2str(figurecount) ', PSD of I Phase Shift & no Noise']);
   figurecount=figurecount+1; %increment figure index
   %time domain plot of in phase signal I with phase shift
   figure (figurecount); %open new figure for plot


   % plot small portion of time domain signal I so that data will fit meaningfully in figure.
   %floor(size(time,2)/scale) selects a small sample of the vectors to plot
   plot (time(1:floor(size(time,2)/scale)),I(1:floor(size(time,2)/scale)));
   title(['Fig # ' num2str(figurecount) ', Time Domain of I with Phase Shift & no Noise']);
   xlabel('{\itTime - Seconds} ');
   ylabel('Amplitude');
   grid on;
   figurecount=figurecount+1;%increment figure index


   %Plot PSD and Time Domain of I+ TX Phase + WGN and Time Domain of I + TX Phase
   figure (figurecount);% open new figure for plot
   psd(IN,[],fs);%plot PSD for specified noise SNR
   title(['Fig # ' num2str(figurecount) ', PSD of I with Phase Shift & Noise SNR='
num2str(10*log10(SNR))]);
   figurecount=figurecount+1;%increment figure index
```

```
    %plot time domain signal I with TX phase shift and WGN at specified SNR
    figure (figurecount);%open new figure for plot
    plot(time(1:floor(size(time,2)/scale)),IN(1:floor(size(time,2)/scale)));
    title(['Fig # ' num2str(figurecount) ', Time Domain of I with Phase Shift & Noise SNR='
num2str(10*log10(SNR))]);
    xlabel('{\itTime - Seconds} ');
    ylabel('Amplitude');
    grid on;
    figurecount=figurecount+1;%increment figure index


    figure (figurecount);% open new figure for plot
    plot(time(1:floor(size(time,2)/scale)),IPWON(1:floor(size(time,2)/scale)));
    title(['Fig # ' num2str(figurecount) ', Time Domain of I with Phase Shift witoutout Noise']);
    xlabel('{\itTime - Seconds} ');
    ylabel ('Amplitude');
    grid on;
    figurecount=figurecount+1;%increment figure index


    % Now check to see if signal is correct by plotting phase shift alone and then determining phase shift
from I+jQ.
    % To determine phase shift, look at the phase angle of I+jQ at every 7th time interval.  Expect to see the
P4 phase
    % function plot repeated 5 times after unwrapping and detrending the phase angle.

    figure(figurecount);%open new figure for plot
    plot(phase);
    title(['Fig # ' num2str(figurecount) ', Phase Shift (radians)']);
    xlabel('i - index for phase change');
    ylabel('Phase Shift - Theta');
    grid on;
    figurecount=figurecount+1;%increment figure index
    %Now strip out points from I and Q to reconstruct phase shift.
    %I(1:SAR:floor(size(I,2)/5)) selects a data points with the phase values corresponding to the original
phase calculation,;
    %by indexing SAR through the first one fifth of the vector computed by floor(size(I,)/5).  The vector was
repeated five times.
    signal=I(1:SAR*k:size(I,2))+j*Q(1:SAR*k:size(I,2));

    phase_signal=angle(signal);%determine the angle from the complex signal

    % unwrap(I) corrects the radian phase angles in array I by adding multiples of ±2pi
    % when absolute jumps between consecutive array elements are greater than pi radians.
    unphase=unwrap(phase_signal);
    figure (figurecount);%open new figure for plot
    plot (unphase);
    title(['Fig # ' num2str(figurecount) ', Phase Shift from I+jQ (radians)']);
    xlabel('i - index for phase change');
    ylabel('Phase Shift - Theta');
    grid on;
    figurecount=figurecount+1;%increment figure index
    x = linspace(0,16,T/Ts); %For scaling the x-axis
    Phased = Phase.*(180/pi); %Enact this line to change into degrees
    figure(figurecount)
    plot(x,Phased)
    title(['Fig # ' num2str(figurecount) ', Phase Shift (degrees)']);
    xlabel('i - index for phase change');
    ylabel('Phase Shift - Theta');
    grid on;
    figurecount=figurecount+1;%increment figure index

    % unwrap corrects the radian phase angles in array I by adding multiples of ±2pi
    % when absolute jumps between consecutive array elements are greater than pi radians.

    % Plot of unwrapped Phase
    unphase2=unwrap(Phased);
    figure (figurecount);%open new figure for plot
    plot(time(1:T/Ts),(phaseunwrapped.*(180/pi)));
    title(['Fig # ' num2str(figurecount) ', Unwrapped Phase']);
    xlabel('Time index');
    ylabel('Phase Shift - Theta');
    grid on;
    figurecount=figurecount+1;%increment figure index


    %Plot of signal to view phase shift
    index4=0;
    Phased = Phase.*(180/pi); %Enact this line to change into degrees
    figure(figurecount)
    for index4=1:T/Ts;
        timeindex(index4)=time(index4);
        Signal(index4)=(I(index4)+j*Q(index4));
    end
    plot(timeindex,Signal)
    title(['Fig # ' num2str(figurecount) ', T' num2str(option2) ' Signal with Phase shift']);
    xlabel('Time index');
    ylabel('Amplitude');

    grid on;
    figurecount=figurecount+1;%increment figure index



else
    disp('Signal not plotted')
    fprintf('\n\n')
end

% This section generates the files for analysis

INP=IN';%transpose I with noise and T1-4 phase shift for text file
QNP=QN';%transpose Q with noise and T1-4 phase shift for text file
IPWONT=IPWON';%transpose I with phase without noise for text file
QPWONT=QPWON';%transpose Q with phase without noise for text file

% % save results in data files
```

```
I=INP(:,1);
Q=QNP(:,1);

II=IPWONT(:,1);
QQ=QPWONT(:,1);

disp(' ')
saveresult = input('Do you want to save the new signal (Y/y or N/n) ?','s');

if (saveresult == 'Y') | (saveresult =='y')
   %determine savingparameters
   x=0;
   switch option2
      case 1
         x=k;
      case 2
         x=k;
      case 3
         x=deltaf;
      case 4
         x=deltaf;
   end

   ff=floor(f/1e3);
   ffs=floor(fs/1e3);
   save(['PT' num2str(option2) '_', num2str(ff), '_', num2str(ffs),...
         '_', num2str(m), '_', num2str(x), '_', num2str(SNR_dB)],'I','Q');
   I=II;
   Q=QQ;
   save(['PT' num2str(option2) '_' num2str(ff) '_' num2str(ffs), ...
         '_', num2str(m), '_', num2str(x), '_s'],'I','Q');
   disp(' ');
   disp(['Signal and noise save as :  PT' num2str(option2) '_' ...
         num2str(ff) '_' num2str(ffs) '_' num2str(m) '_' num2str(x) ...
         '_' num2str(SNR_dB)]);
   disp(['Signal only save as :         PT' num2str(option2) '_' ...
         num2str(ff) '_' num2str(ffs) '_' num2str(m) '_'...
         num2str(x)  '_s']);
   disp(['Directory:                        ' num2str(cd)]);
else
   disp(' ')
   disp('Signal not saved')
   fprintf('\n\n')
end
```

APPENDIX B

Code developed in MATLAB to pre-process images before classification and to generate the training vectors.

```
%===========================================================================
% Signal matrices FOR TRAINING/TEST SIGNALS
% By Christer Persson
% Last modified AUGUST 30, 2003
% Code to generate the matrices describing training signals
%===========================================================================

clear
clear all
close all

% Generation of matrix that describes the training signals for separate
% networks, Signal only
Msignal=[1 1 1 1 0 0 0 0 0 0 0 0 0 0 0     % BPSK
%        0 0 0 0 0 0 0 0 0 0 0 0 0 0 0     % Costas
         0 0 0 0 1 1 1 1 0 0 0 0 0 0 0     % FMCV
         0 0 0 0 0 0 0 0 1 1 0 0 0 0 0     % Frank
%        0 0 0 0 0 0 0 0 0 0 0 0 0 0 0     % P1
%        0 0 0 0 0 0 0 0 0 0 0 0 0 0 0     % P2
%        0 0 0 0 0 0 0 0 0 0 0 0 0 0 0     % P3
         0 0 0 0 0 0 0 0 0 0 1 1 0 0 0     % P4
         0 0 0 0 0 0 0 0 0 0 0 0 0 0 0];   % PT1
%        0 0 0 0 0 0 0 0 0 0 0 0 0 0 0     % PT2
%        0 0 0 0 0 0 0 0 0 0 0 0 0 0 0     % PT3
%        0 0 0 0 0 0 0 0 0 0 0 0 0 1 1];   % PT4
save('Matrices\Msignal','Msignal');

% Generation of matrix that describes the training signals for total
% network, Signal only
MtotalSignal=[1 1 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0     % BPSK
%             0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0     % Costas
              0 0 0 0 0 0 1 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0     % FMCV
              0 0 0 0 0 0 0 0 0 0 0 1 1 1 0 0 0 0 0 0 0 0     % Frank
%             0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0     % P1
%             0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0     % P2
%             0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0     % P3
              0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 0 0 0 0     % P4
              0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0];   % PT1
%             0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0     % PT2
%             0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0     % PT3
%             0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1];   % PT4
save('Matrices\MtotalSignal','MtotalSignal');

% Generation of matrix that describes the training signals for separate
% networks, Signal plus noise
```

```
Mnoise=[1 1 1 1 1 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0    % BPSK
%       0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0    % Costas
     0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0    % FMCV
     0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0    % Frank
%       0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0    % P1
%       0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0    % P2
%       0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0    % P3
     0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 0 0 0 0 0 0    % P4
     0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0];    % PT1
%       0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0    % PT2
%       0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0    % PT3
%       0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1];   % PT4
save('Matrices\Mnoise','Mnoise');

% Generation of matrix that describes the training signals for the combined
% network
Mtotal=[1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0   % BPSK
%       0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0    % Costas
     0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0   % FMCV
     0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0   % Frank
%       0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0    % P1
%       0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0    % P2
%       0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0    % P3
     0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 0 0 0 0 0 0
0    % P4
     0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0];    % PT1
%       0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0    % PT2
%       0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0    % PT3
%       0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1
1 1 1];   % PT4
save('Matrices\Mtotal','Mtotal');


%=====================================================================
% PREPROCESSING FOR SIGNALS
% By Christer Persson
% Last modified AUGUST 30, 2003
% Code to preprocess all training signals, noise + "pure"
%=====================================================================

clear
clear all
close all

% default values for image cropping and structuring element
x1=[75 50 255 255];     % size image crop WD
```

```
x2=[190 150 127 127];     % size image crop CYCL
x3=[75 110 255 255];    % size image crop QMFB

% Assigning structuring element for image processing
se1=strel('diamond',4);
se2=strel('line',5,-30);
se3=strel('line',5,30);
se4=strel('line',5,-60);
se5=strel('line',5,60);
se6=strel('line',5,90);
se7=[1 0 0
    0 1 0
    0 0 1];

se8=[0 0 1
    0 1 0
    1 0 0];

se9=0-[0 0 0 1 1 0 0
      0 0 0 1 1 0 0
      0 0 0 1 1 0 0
      0 0 0 1 1 0 0
      0 0 0 1 1 0 0
      0 0 0 1 1 0 0];

se10=[0 0 0 0 1 1 1 0 0
     0 0 0 1 1 1 0 0 0
     0 0 1 1 1 1 1 0 0
     0 1 1 0 0 0 1 1 0
     1 1 0 0 0 0 0 1 1
     1 0 0 0 0 0 0 0 1];

se11=[1 1 0 0 1 0 0 1 1
     0 1 1 0 1 0 1 1 0
     0 0 0 1 1 1 1 0 0
     0 0 1 1 1 1 1 0 0
     0 1 1 0 1 0 1 1 0
     1 1 0 0 1 0 0 1 1];

% Getting number of files in signal directories
dWD=dir('C:\Documents and Settings\Christer\My Docu-
ments\Thesis\FinalWork\Preprocessing\WDimages\WDall');
dCYCL=dir('C:\Documents and Settings\Christer\My Docu-
ments\Thesis\FinalWork\Preprocessing\CYCLimages\CYCLall');
dQMFB=dir('C:\Documents and Settings\Christer\My Docu-
ments\Thesis\FinalWork\Preprocessing\QMFBimages\QMFBall');
ltWD=length(dWD);
ltCYCL=length(dCYCL);
ltQMFB=length(dQMFB);


%-------------------------------------------------------------------------
% WD Preprocessing
%-------------------------------------------------------------------------

% Running the loop to get all signal files
```

```
for i1=3:(ltWD-1)   % From 3 to avoid directories

   % Reading image, converting to grey scale, crop to get area of interest
   [NWD,map1]=imread(['C:\Documents and Settings\Christer\My Docu-
ments\Thesis\FinalWork\Preprocessing\WDimages\WDall\',dWD(i1).name],'bmp');
   NWD1=ind2gray(NWD,bone);

   % Cropping image to region of interest
   NWD2=1*imcrop(NWD1,x1);

   % Adjusting image intensity to fill whole range
   NWD3 = imadjust(NWD2, stretchlim(NWD2), [0 1]);

   % Adaptive Filtering
   NWD3=wiener2(NWD2,[5 5]);

   % Filtering to extract features
   NWD4=imtophat(NWD3,se10);
   NWD5=imbothat(NWD3,se10);
   NWD6=imsubtract(NWD5,imadd(NWD4,NWD3));

   % Dilation of image
   NWD61=imdilate(NWD6,se11);
   NWD62=imdilate(NWD61,se1);

   % Adjusting image intensity to fill whole range
   NWD7 = imadjust(NWD62, stretchlim(NWD62), [0 1]);

   % Closing
   NWD8=imclose(NWD7,se1);
   NWD9=imclose(NWD8,se2);
   NWD10=imclose(NWD9,se3);
   NWD11=imclose(NWD10,se4);
   NWD12=imclose(NWD11,se5);
   NWD13=imclose(NWD12,se6);
   NWD14=imadjust(NWD13, stretchlim(NWD13), [0 1]);

   % Doing distance transform
   NWD15=bwdist(NWD14,'quasi-euclidean');

   % Saving image to folder for inspection
   imwrite(NWD15,['C:\Documents and Settings\Christer\My Docu-
ments\Thesis\FinalWork\Preprocessing\Processimages\WDimages\',dWD(i1).name],'bmp');

   % Reshaping to fit Neural Network
   NWD16=imresize(NWD15,[16 32],'bilinear');
   NWD17=reshape(NWD16,512,1);
   WDtrainingAll(:,(i1-2))=1*NWD17;
end

% Saving generated matrix
save('Matrices\WDall','WDtrainingAll');


%-----------------------------------------------------------------------
% CYCL Preprocessing
```

```
%------------------------------------------------------------------------

% Running the loop to get all signal files
for i2=3:(ltCYCL-1);

   % Reading image, converting to grey scale, crop to get area of interest
   [NCYCL,map2]=imread(['C:\Documents and Settings\Christer\My Docu-
ments\Thesis\FinalWork\Preprocessing\CYCLimages\CYCLall\',dCYCL(i2).name],'bmp');
   NCYCL1=ind2gray(NCYCL,bone);

   % Cropping image to region of interest
   NCYCL2=1*imcrop(NCYCL1,x2);

   % Adjusting image intensity to fill whole range
   NCYCL3 = imadjust(NCYCL2, stretchlim(NCYCL2), [0 1]);

   % Adaptive Filtering
   NCYCL3=wiener2(NCYCL2,[5 5]);

   % Filtering to extract features
   NCYCL4=imtophat(NCYCL3,se10);
   NCYCL5=imbothat(NCYCL3,se10);
   NCYCL6=imsubtract(NCYCL5,imadd(NCYCL4,NCYCL3));

   % Dilation of image
   NCYCL61=imdilate(NCYCL6,se11);
   NCYCL62=imdilate(NCYCL61,se1);

   % Adjusting image intensity to fill whole range
   NCYCL7 = imadjust(NCYCL62, stretchlim(NCYCL62), [0 1]);

   % Dilation
   NCYCL8=imclose(NCYCL7,se1);
   NCYCL9=imclose(NCYCL8,se2);
   NCYCL10=imclose(NCYCL9,se3);
   NCYCL11=imclose(NCYCL10,se4);
   NCYCL12=imclose(NCYCL11,se5);
   NCYCL13=imclose(NCYCL12,se6);
   NCYCL14=imadjust(NCYCL13, stretchlim(NCYCL13), [0 1]);

%    % Doing distance transform
%    NCYCL15=bwdist(NCYCL14,'quasi-euclidean');

   % Saving image to folder for inspection
   imwrite(NCYCL14,['C:\Documents and Settings\Christer\My Docu-
ments\Thesis\FinalWork\Preprocessing\Processimages\CYCLimages\',dCYCL(i2).name],'bmp');

   % Reshaping to fit Neural Network
   NCYCL16=imresize(NCYCL14,[16 32],'bilinear');
   NCYCL17=reshape(NCYCL16,512,1);
   CYCLtrainingAll(:,(i2-2))=1*NCYCL17;
end
% Saving generated matrix
save('Matrices\CYCLall','CYCLtrainingAll');
```

```
%------------------------------------------------------------------------
% QMFB Preprocessing
%------------------------------------------------------------------------

% Running the loop to get all signal files
for i3=3:(ltQMFB-1)

   % Reading image, converting to grey scale, crop to get area of interest
   [NQMFB,map3]=imread(['C:\Documents and Settings\Christer\My Docu-
ments\Thesis\FinalWork\Preprocessing\QMFBimages\QMFBall\',dQMFB(i3).name],'bmp');
   NQMFB1=ind2gray(NQMFB,map3);

   % Cropping image to region of interest
   NQMFB2=1*imcrop(NQMFB1,x3);

   % Adjusting image intensity to fill whole range
   NQMFB3 = imadjust(NQMFB2, stretchlim(NQMFB2), [0 1]);

   % Adaptive Filtering
   NQMFB3=wiener2(NQMFB2,[5 20]);
   NQMFB34 = imadjust(NQMFB3, [0 0.8], [0 1]);

   % Filtering to extract features
   NQMFB4=imtophat(NQMFB34,se10);
   NQMFB5=imbothat(NQMFB34,se10);
   NQMFB6=imsubtract(NQMFB5,imadd(NQMFB4,NQMFB34));

   % Dilation of image
   NQMFB61=imdilate(NQMFB6,se11);
   NQMFB62=imdilate(NQMFB61,se1);

   % Adjusting image intensity to fill whole range
   NQMFB7 = imadjust(NQMFB62, stretchlim(NQMFB62), [0 1]);

   % Dilation
   NQMFB8=imclose(NQMFB7,se1);
   NQMFB9=imclose(NQMFB8,se2);
   NQMFB10=imclose(NQMFB9,se3);
   NQMFB11=imclose(NQMFB10,se4);
   NQMFB12=imclose(NQMFB11,se5);
   NQMFB13=imclose(NQMFB12,se6);
   NQMFB14=imadjust(NQMFB13, stretchlim(NQMFB13), [0 1]);

%    % Doing distance transform
%    NQMFB15=bwdist(NQMFB14,'quasi-euclidean');

   % Saving image to folder for inspection
   imwrite(NQMFB14,['C:\Documents and Settings\Christer\My Docu-
ments\Thesis\FinalWork\Preprocessing\Processimages\QMFBimages\',dQMFB(i3).name],'bmp');

   % Reshaping to fit Neural Network
   NQMFB16=imresize(NQMFB14,[16 32],'bilinear');
   NQMFB17=reshape(NQMFB16,512,1);
   QMFBtrainingAll(:,(i3-2))=1*NQMFB17;
end
```

```
% Saving generated matrix
save('Matrices\QMFBall','QMFBtrainingAll');
```

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF REFERENCES

[1] Phillip E. Pace, Notes for EC 4690 (Network Centric Radar Electronic Warfare: Techniques and Systems for International Students), Naval Postgraduate School, 2003 (unpublished).

[2] D. Curtis Schleher, "Low Probability of Intercept Radar," *IEEE Iternational Radar Conference,* pp. 346-349, August 1985.

[3] Fernando Taboada, "Detection and Classification of LPI Radar Signals Using Parallel Filter Arrays and Higher Order Statistics," Master's Thesis, Naval Postgraduate School, Monterey, California, 2002.

[4] Antonio F.Lima, Jr., "Analysis of low probability of intercept (LPI) radar signals using cyclostationary processing," Master's Thesis, Naval Postgraduate School, Monterey, California, 2002.

[5] Jen Y. Gau, "Analysis of LPI Radar Signals Using Wigner Distribution," Master's Thesis, Naval Postgraduate School, Monterey, California, 2002.

[6] Pedro Jarpa, "Quantifying the Differences in Low Probability of Intercept Radar Waveforms Using Quadrature Mirror Filtering," Master's Thesis, Naval Postgraduate School, Monterey, California, 2002.

[7] Pace E. Phillip, *Detecting and Classifying Low Probability of Intercept Radar*, 1st. ed., Artech House, Norwood, Massachusetts, 2003.

[8] John E. Fielding, "Polytime Coding as a Means of Pulse Compression," *IEEE Trans. On Aerospace and Electronic Systems,* Vol. 35, No. 2, pp. 716-721, 1999.

[9] Sergios Theodoridis and Konstantinos Koutroumbas, *Pattern Recognition*, 1st ed. pp. 139-232. Academic Press, San Diego, 1999.

[10] Christopher M. Bishop, *Neural Networks for Pattern Recognition,* 1st. ed. pp. 297-329. Oxford University Press, New York, 1995.

[11] Sergio Barbarossa and Olivier Lemoine, "Analysis of non-linear FM signals by pattern recognition of their time–frequency representation," *IEEE Signal Processing Letters,* Vol. 3, No. 4, p. 112, April 1996.

[12] Edith Grall-Maës and Pierre Beauseroy, "Mutual information-based feature extraction on the time–frequency plane," *IEEE Trans. on Signal Processing,* Vol. 50, No. 4, pp. 779-790, April 2002.

[13] Rafael C. Gonzales and Richard E. Woods, *Digital Image Processing*, 2nd ed. pp. 519-642. Prentice-Hall, Upper Saddle River, New Jersey, 2002.

[14] The MathWorks Inc., "*Detecting Image Processing Toolbox*," Version 3.0, The MathWorks Inc., Natick, Massachusetts, 2001.

[15] C. Neubauer and M. Fang, "Performance comparison of feature extraction methods for neural network based object recognition," *Proc. of the 2002 International Joint Conference on Neural Networks, IJCNN '02,* Vol. 2, pp. 1608-1613, IEEE, New York, IEEE Cat. No. 02CH37290, May 2002.

INITIAL DISTRIBUTION LIST

1. Defense Technical Information Center
   Ft. Belvoir, Virginia

2. Dudley Knox Library
   Naval Postgraduate School
   Monterey, California

3. Chairman, Code EC
   Department of Electrical and Computer Engineering
   Naval Postgraduate School
   Monterey, California

4. Chairman, Code IW
   Department of Information Sciences
   Naval Postgraduate School
   Monterey, California

5. Prof. Phillip E. Pace
   Center for Joint Services Electronic Warfare
   Naval Postgraduate School
   Monterey, California

6. Prof. D. Curtis Schleher
   Department of Information Sciences
   Naval Postgraduate School
   Monterey, California